Fall 12-17-2021

Privacy Preserving for Multiple Computer Vision Tasks

Amala Varghese Wilson

Privacy Preserving for Multiple Computer Vision Tasks

A Project

Presented to

Dr. Melody Moh

Dr. Mashhour Solh

Dr. Nada Attar

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Amala Varghese Wilson

December 2021

ABSTRACT

Privacy-preserving visual recognition is an important area of research that is gaining momentum in the field of computer vision. In a production environment, it is critical to have neural network models learn continually from user data. However, sharing raw user data with a server is less desirable from a regulatory, security and privacy perspective. Federated learning addresses the problem of privacy-preserving visual recognition. More specifically, we closely examine and dissect a framework known as Dual User Adaptation (DUA) presented by Lange et al. at CVPR 2020, due to its novel idea of bringing about user-adaptation on both the server-side and user device side. Data in the server and user device is predefined into a series of tasks prior to training and testing. However, since user data is constantly evolving, it's important to see how DUA performs on unseen data or tasks. A few implementations are also executed to see if the performance of the DUA model can be improved on unseen data. In addition, two other federated learning frameworks are implemented to compare how it performs with DUA. Through this research we show that retraining the classifier layer of the merged model with all data categories greatly improves the performance for real-world implementation of DUA.

Keywords – Privacy-preserving, Federated Learning, Dual User-Adaptation, FedAvg, FedProx

ACKNOWLEDGMENTS

I would like to take some time to acknowledge and thank the people who have been with me from the start of the project to the end.

Firstly, I would like to thank my main advisor, Dr. Mashhour Solh, for guiding me throughout this project. Aside from his vast knowledge in deep learning and computer vision, he always paved the path to help me think critically which helped me get a better understanding of the topic itself. Secondly, I would like to thank Dr. Melody Moh, for the continuous guidance. I would also like to thank Dr. Nada Attar for the feedback she provided and support. Alongside with the professors, I would also like to acknowledge my family for being a constant support to me throughout these months because I wouldn't have been able to do it without them.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The ubiquity of smart devices, such as security cameras, phones, and watches, in today's technology-driven world has made people's lives convenient. For example, lifestyle and fitness apps have been incredibly adopted during the COVID-19 pandemic lockdowns [1]. These apps commonly rely on deep learning models as their primary engines. The availability of large high-quality datasets is critical to train deep learning models, and in a production environment, it is especially important to have the model continually learn new tasks from streams of user data to maintain the relevancy and performance of the model and the app [2]. Additionally, user data could potentially reduce huge investments that go into building a manually curated and labeled dataset for training models, and this is especially useful in scenarios where most of the data is housed within data islands [3]. However, it is not always in the interest of the user to send raw user data to a central server due to privacy concerns and vulnerability of data leaks due to cyberattacks [4]. Thus, development of privacy-preserving frameworks for continual and user-personalized learning is of utmost importance.

Privacy-preserving continual learning was popularized in a large scale by the introduction of federated learning into Google's Android keyboard in 2017 [5]. Since then, numerous frameworks have been proposed and deployed, with some designed for domain-specific applications [6]. Federated learning frameworks can be broadly classified into two categories based on whether a central server with significant compute capability is used as a manager to collect deidentified user data (in the form of

prediction target probabilities or model weights) and perform aggregation and training. A federated learning framework with such a powerful central server is useful while developing models that need to be deployed on low-powered devices and for comparatively less-sensitive user data. However, a central server can be less attractive to users when the data involves sensitive information such as health records or financial information. In such cases, a decentralized federated learning framework is more suitable. Recent examples of centralized federated learning frameworks include FedAvg [7], FedSVRG [8], Agnostic FL [9] whereas decentralized federated learning frameworks include SimFL [10], Swarm Learning [11], and Galaxy Federated Learning [12].

The Dual User Adaptation (DUA) framework was a solution developed by Lange et al. that closely resembles federated learning [13]. The authors describe a highly scalable continual learning centralized federated learning framework that 1) avoids sending raw user data to the central server, and 2) returns a final model that is user-personalized and not just a server model that learns general trends from data obtained from a pool of users [13]. To protect users' privacy, the authors took an unsupervised approach to personalize models with unlabeled local user images. The authors tested the DUA framework for image classification application on the MNIST [19] and SVHN [20] and MIT indoor scenes [21] datasets. However, the DUA framework is domain-agnostic and should, in theory, be suitable for a wide range of applications. Our research builds upon the DUA framework by extending its capability to deal with unseen user data through

the following steps: (1) a thorough investigation of the DUA framework, (2) evaluating server trained (unmerged) and merged models on unseen data, (3) implementing different model configurations to train and test merged models and evaluate which one performed the best on unseen data. Through this research we show that retraining the classifier layer of the merged model with all data categories greatly improves the performance for real-world implementation of DUA.

The paper is structured as follows. Section II provides an overview of related work done on privacy-preserving visual recognition. Section III provides background information on the DUA framework in detail. Subsequently, Section IV provides a description of the research objective. Section V provides a review of the DUA framework. Section VI provides information on the preliminary research. Section VII provides information on the methodology that was taken to meet the research objective. Section VIII describes the evaluation results of the experiments that were executed. Lastly, Section IX recounts this research and sets up the building blocks for future work. This report also includes a supplementary section that provides additional information on this research project.

II. RELATED WORK

Privacy-preserving visual recognition has become an important area of research in the field of computer vision as more and more deep neural network models are being trained on private, sensitive data. Many researchers have and continue to explore this

area of research. As a result, varied solutions have been developed to tackle the problem of privacy-preserving visual recognition. The adversarial training framework proposed by Wu et al. [14] is an example of one such solution. Similarly, Papernot et al., developed a framework known as Private Aggregation Teacher Ensembles (PATE) [29]. Zhu et al. used PATE as the foundation for their research by focusing on the most important parameter in PATE and proposed a different algorithm [30]. FedAvg [7] and FedProx [15] are two federated learning frameworks that also address this issue.

Adversarial Training Framework: The motivation for the study conducted by Wu et al. came from a growing increase of privacy concern due to the prevalence of smart surveillance systems [14]. Videos and images captured by these smart devices had to be uploaded to a centralized cloud server to perform backend analytics to provide enhanced and tailored user experiences [14]. Cryptographic solutions were not sufficient to prevent attackers from accessing this data. In addition, users' privacy was compromised when authorized analysts mined the data to gather important information [14]. As the first step towards solving the dilemma of providing user convenience while still protecting the user's privacy, Wu et al. incorporated the concept of differential privacy in the adversarial training framework [14]. The goal of the adversarial training framework was to optimize target task performance without compromising users' data by learning an "active degradation" or transform that can be applied to raw visual data [14].

1) *Brief Overview of Differential Privacy*: Differential privacy is a technique that has

been employed by many businesses to keep user data private while collecting and analyzing user data to improve user experience. This technique involves applying statistical functions to data to anonymize the data for the purpose of protecting the data [16]. Applying statistical functions to data inserts random noise to the data making it more secure than if the data was simply sent as a response to a query in its raw, original format. However, one of the challenges that is present in differential privacy is knowing how much noise to add to data [16]. In other words, the amount of noise is a trade-off. The more noise that is added to data, the more anonymous that data becomes, but also makes the data less useful. One of the ways in which differentially private systems try to enforce a privacy guarantee is by enforcing a maximum privacy loss, known as the privacy budget.

2) *Technical Approach/Methodology:* Fig. 1 represents the model architecture diagram of the proposed framework.

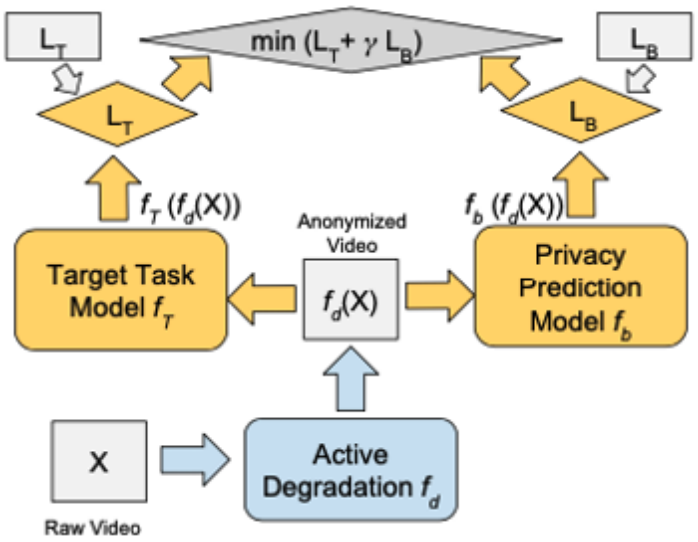


Figure 1: Adversarial Training Framework [taken from Wu et al. (2020)]

$$\min_{f_T, f_d} L_T(f_T(f_d(X), Y_T) + \gamma \max_{f_b \in \mathcal{P}} L_B(f_b(f_d(X)), Y_B)$$

Equation 1: Mathematical Equation of Adversarial Training Framework
[taken from Wu et al. (2020)]

The raw video data, *X,* is first fed into the model and passes through the active degradation function f_d producing the anonymized video $f_d(X)$. The anonymized video is passed to two models, the target task model, $f_T(f_d(X))$, and privacy prediction model, $f_b(f_d(X))$, simultaneously during training. The target task model, $f_T(f_d(X))$, was implemented to predict the target task it was trained on. For example, in this study, the target task was classifying human action. The privacy prediction model, $f_b(f_d(X))$, was implemented to evaluate how well the model performs on identifying private information from the data. The output of the target task model, $f_T(f_d(X))$, and labels of training data, Y_T, are passed to the target task cost function L_T to get a measure of how well the model performed on the transformed raw data. The output of the privacy prediction model, $f_b(f_d(X))$, and labels of "privacy-related annotations" [14], Y_B, are passed to the privacy budget cost function, L_B, to get a measure of how well privacy has been preserved [14]. The mathematical equation of this framework can be found in Equation 1. In this equation, the weight parameter is represented by γ and P represents the family of privacy prediction functions. Based on Fig. 1 and Equation 1, the entire model is trained end-to-end under the hybrid loss of L_T

and L_B with the intention of maximizing target task performance while minimizing privacy breach.

Private k-Nearest Neighbors: Many machine learning algorithms have been trained using private, sensitive data so they can be used to produce applications best tailored to its users. However, there is no guarantee that the trained data will remain private. In fact, overfitting is a prominent issue for many algorithms [29]. This implicit memorization enables attackers to gain unauthorized access to private information that can lead to unwanted situations.

According to Papernot et al., there is a direct and indirect way to attack machine learning models. The direct approach is by "analyzing model parameters [29]" whereas the indirect approach is to repeatedly query models to gather as much data as possible. To protect the privacy of training data, Papernot et al., developed a structured framework based off the idea of teacher student transfer technique known as Private Aggregation Teacher Ensembles (PATE) [29]. In this framework, as seen in Fig. 2, sensitive data is divided into a few disjoint subsets which are fed into teacher models to train them. The resulting output of the teacher models and unlabeled data is then used to train a student model. With this approach, even if attackers are aware of the student model's internal parameters, the privacy of the training data will be protected since the student model won't depend on a single training data point [29].

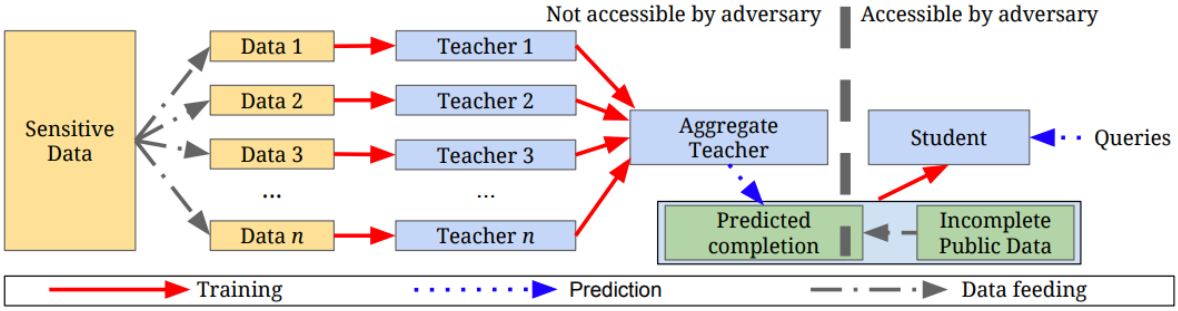


Figure 2: PATE [taken from Papernot et al. (2017)]

The key idea behind PATE is to guarantee the privacy of the data by limiting the access students have to their teachers. For this purpose, Papernot et al. adopted generative adversarial networks (GANs) to speed up the knowledge transfer between student and teacher [29]. Zhu et al. observed that the parameter *k,* which represents the number of disjoint teachers, is the most important parameter in PATE [30]. In addition, according to Zhu et al., if the teacher model is a deep neural network model, then large amounts of labeled data is required to obtain high performance [30]. However, obtaining labeled data is an expensive task and choosing a large enough value for *k* would be insufficient since it will only generate a very small subset of data. To tackle this problem, Zhu et al. proposed an algorithm called Private k-Nearest Neighbors (Private kNN) [30].

Federated Averaging: Federated learning is a term developed by McMahan et al. as part of their research in providing a decentralized approach to training models by using user data without compromising the privacy of the data [7]. In a traditional setting, user data can be sent to a central server to train models which in turn will

produce high performing and user personalized models. However, this poses a threat to the privacy of the user data. In federated learning, a single server model is shared among all user devices where the model gets trained on local user data [7]. After the training is complete, the locally trained models are sent back to the server and aggregated into a global model, which is again sent back to the user device for further training and this cycle is repeated continuously.

The Federated Averaging (FedAvg) algorithm was developed by McMahan et al. as part of their research work in federated learning [7]. A diagram of the FedAvg algorithm can be seen in Fig. 3. As seen in Fig. 3, this algorithm assumes a setup of a central server and a total of K users interacting with the central server [7]. In addition, each user j has a fixed local dataset P_j. To begin with, the central server initializes the weights of the shared global model randomly. Next, a random fraction C of the total number of users K is taken and the global model is sent out to each of these C users. Once each user has received the global model parameters, each selected user performs training on local data and sends the updated model parameters back to the server. After the server receives the updates, the global model gets updated by averaging all the user updates. This process continues for multiple rounds.

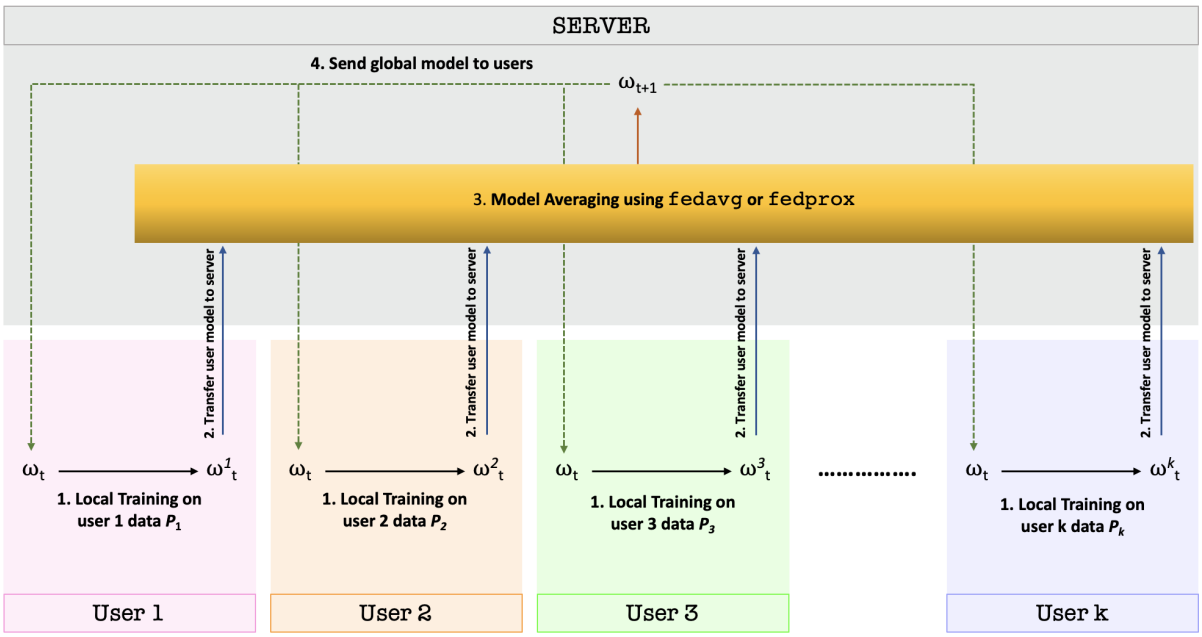


Figure 3: Federated Learning Frameworks – FedAvg [McMahan et al.] & FedProx [Li et al.]

FedProx: Similar to FedAvg [7], FedProx is another solution for federated learning [15]. Federated learning comes with its own set of challenges. The first challenge is associated with the variable of system architecture in user devices [15]. For example, there is no guarantee that the devices that are connected to a central server are going to be composed of the same system architecture. This is an important attribute to keep in mind when developing a federated learning solution. The second challenge is related to the data distributed across multiple user devices [15]. The data contained in user devices can vary greatly. There is a high probability that data across multiple user devices are non-identical. Li et al developed FedProx to address these key challenges [15].

FedProx [15] is closely related to FedAvg [7]. Fig. 3 is also adapted to FedProx. Just like FedAvg, a fraction of the total number of devices are selected at random. The server sends the global model to each of the devices in the subset. Training occurs on the local data and the devices send back the updates to the server where everything is averaged into the global model. The difference between FedProx and FedAvg is that in FedProx, each device does a different amount of work depending on its system architecture to account for system heterogeneity.

III. BACKGROUND INFORMATION

The Dual User Adaptation (DUA) framework was developed by Lange et al. as an innovative solution to the problem of privacy-preserving visual recognition [13]. Closely resembling federated learning, this framework aims to preserve the privacy of raw user data while still delivering personalized models to the user.

In the DUA framework, unlike in traditional federated learning, user-personalization occurs both on the server and user device, which is defined by two adaptation functions, ψ and ϕ, respectively. The DUA framework can be broken down into two phases. Server S contains a set of N task-specific models, $M = \{M_1, M_2, ..., M_N\}$, as seen on the right-hand side of Fig. 4.

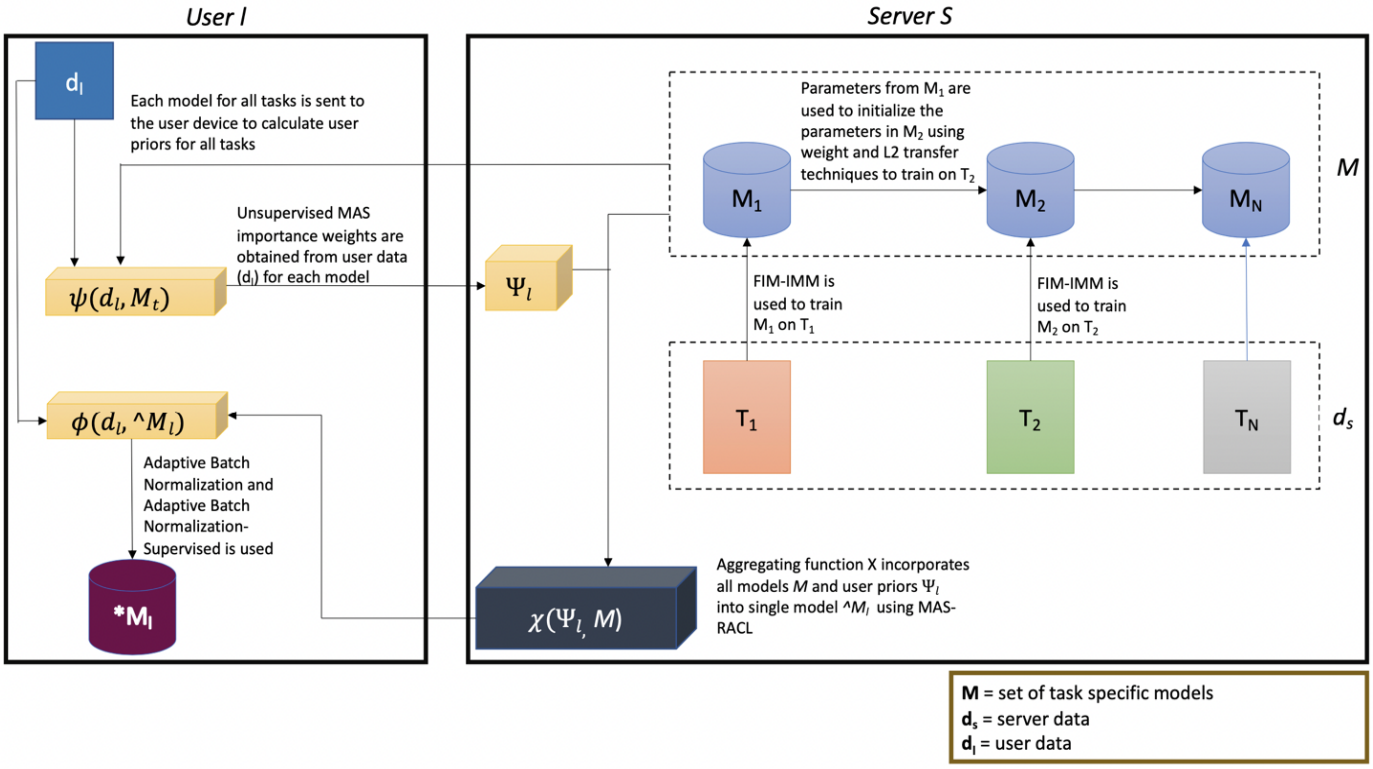


Figure 4: Unsupervised Model Personalization [adapted from Lange et al (2020)]

In the first phase, these models are trained sequentially using task incremental learning from the labeled data d_S [13]. Each model in M is dependent upon the current task data and previous task model, with each subsequent model having its weights initialized at the start of training to the weights of the model for the previous task. Since the DUA framework is designed for continual learning, the server learns a new task T_n with corresponding new task data D_n after which D_n is discarded and only the model M_n is kept [13]. When a model is trained sequentially, there is a risk that the model may forget the information it last learned on the previous task it was trained for when the

model is learning a new task. This risk is coined the term catastrophic forgetting [17]. To address the issue of catastrophic forgetting, the server adopts a model adaptation strategy known as Incremental Moment Matching (IMM) to train [17]. In statistics, moments can be described as a robust way of describing a dataset. There are varying degrees of moments. For instance, the first moment is the mean of a dataset, the second moment is the average squared distance from 0 of a dataset, and the third moment is the variance of a dataset. Combining Bayesian neural networks and moments in statistics, IMM resolves the issue of catastrophic forgetting. IMM is a method developed by Lee et al. that uses Gaussian posterior to train sequential models through the method of weight transfer [17]. Mean-IMM is another function of IMM that takes the average of the parameters of two or more models and is utilized in the merging process [17]. In the second phase, the task specific models are used to gather priors, or importance weights, from user data. Then, both task specific models and importance weights are aggregated to a model using the aggregation function, χ . As seen on the bottom left-hand side of Fig. 4, user l receives $\hat{M}_l$ from the server S where the local adaptation function ϕ is applied to the model to get a final model $*M_l = \phi(d_l, \hat{M}_l)$. Since the user devices are resource-limited, training procedures on these devices have been restricted to those requiring very low computational demands.

IV. RESEARCH OBJECTIVE

The differentiating factor between the DUA framework and other federated learning frameworks is that the DUA framework considers user-adaptation both on the server side and user device side. After thoroughly examining the DUA framework and the training process provided by the authors [13], I've identified several shortcomings which I present in detail in Section V. For instance, it was found that the same dataset was used to train the models in the server and served as user data. Hence, when important features were extracted from the user data and sent to the server to merge models, it doesn't ensure that the models work for unseen data. In this research, unseen data is defined as data on the user device that constitutes a task or set of tasks that were not used for model training on the server side. The purpose of this research is to address this issue and improve the performance of merged models. In addition, while there were two experiments performed on the MIT indoor scenes dataset [21], there was no baseline to compare the DUA with in the original publication. For this purpose, FedAvg [7] and FedProx [15] were implemented to check how well DUA performed on this task.

A. Challenges and Innovative Aspects of the Research

- The main challenge of this research is identifying the pitfalls of the DUA framework and introducing improvements to make it robust for unseen data. The DUA framework incorporates many complex concepts:
    - Catastrophic Forgetting [17]

- Incremental Moment Matching (IMM) [17]
- Bayesian Inference [23]
- Prior and Posterior Distribution [23]
- Gaussian Distribution [24]
- Task Incremental Learning [25]
- Continual Learning [26]
- Fisher Information [27]
- Memory Aware Synapses [22]

V. REVIEW OF DUA FRAMEWORK

A. *Experiments Conducted on DUA Framework*

Unsupervised adaptation, scalability and privacy preserving are the three key features of the DUA framework [13]. To demonstrate these three key features, a set of three experiments were performed by Lange et al. on image classification [13]. The first experiment was performed on the MNIST [19] and SVHN [20] datasets. The last two experiments were performed on the MIT Indoor Scenes dataset [21].

1) Numbers Experiment: The first experiment involved incrementally training a Multilayer Perceptron (MLP) model, which consisted of 2 hidden layers of 100 units each on a series of tasks defined by a combination of the MNIST and SVHN datasets [13]. Both these datasets are composed of digits that range from 0 to 9. Taking this into account, a total of 5 tasks were defined for this experiment and each task is

composed of two digits. Each task is responsible for classifying the specified subset of digits it contains. Task 1 consists of digits 0 and 1, task 2 consists of digits 2 and 3, task 3 consists of digits 4 and 5, task 4 consists of digits 6 and 7, and lastly, task 5 consists of digits 8 and 9. Subsequently, each task has corresponding images from both MNIST and SVHN datasets. For example, task 1 has images from both MNIST and SVHN that contain 0 and 1. A clear picture of the defined tasks can be found in Table I.

TABLE I: Tasks of Numbers Experiments

TASKS	SUBSET OF DIGITS
1	0, 1
2	2, 3
3	4, 5
4	6, 7
5	8, 9

In this experiment, the training dataset from both MNIST and SVHN was parsed into the defined tasks in Table I and used for training the models sequentially on server *S*, based on the setup showcased in Fig. 4. To address the feature of scalability, this experiment considered two users. The first user prefers SVHN, therefore, the data in this user device only contains the testing dataset from SVHN parsed into the same tasks as defined in Table I. The second user prefers MNIST, therefore, the data in this user device only contains the testing dataset from MNIST parsed into the same tasks as defined in Table I.

a) *Server Training of Models for all Tasks:* Complying with the DUA framework, these tasks are trained incrementally on the server using IMM [13]. This experiment starts with a MLP model with 10 output nodes as the base model, as shown in Fig. 5.

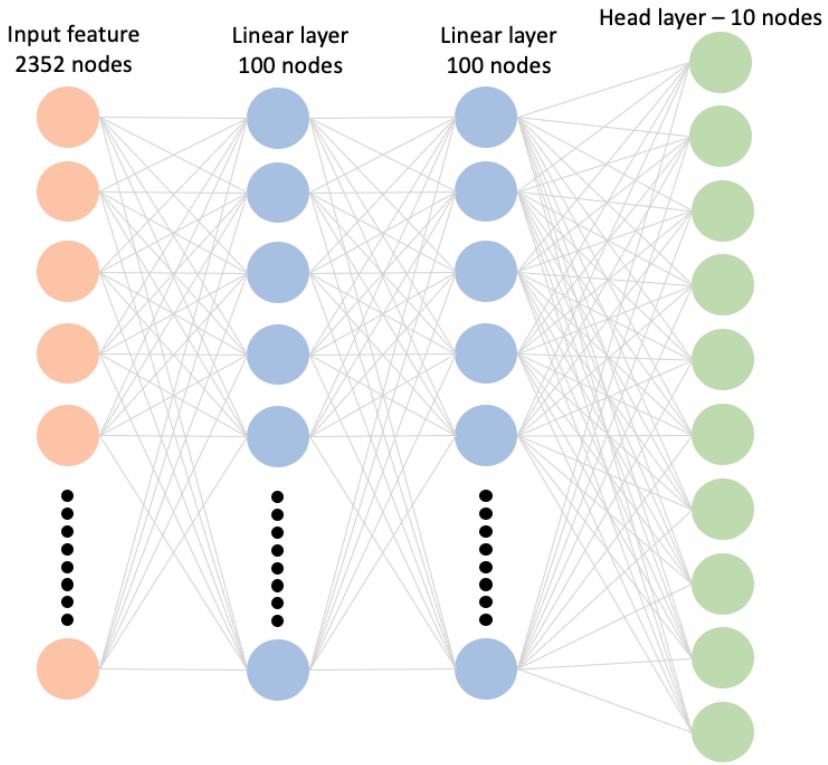


Figure 5: Multilayer Perceptron (MLP) Model with 2 hidden layers of 100 nodes each and 10 output nodes

It's important to note that task 1 is trained differently than all subsequent tasks because the base model has not been trained on any data yet. Since task 1 is only expecting two outputs, [0,1], the last layer, also known as the classifier layer or head layer, of the base MLP model is replaced with 2 output nodes instead of 10, as shown in Fig. 6.

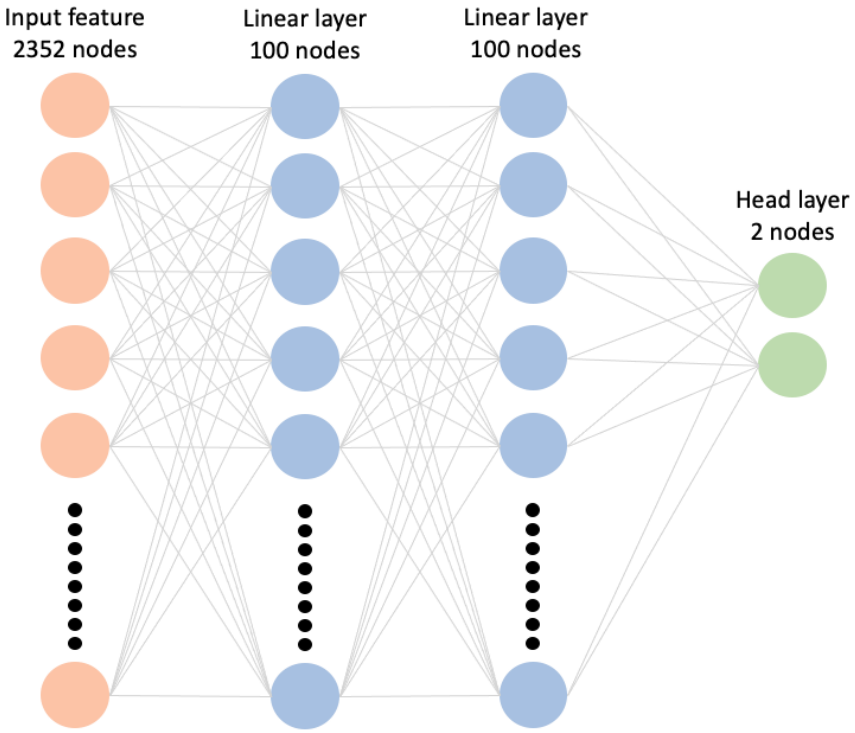


Figure 6: Multilayer Perceptron (MLP) Model with 2 hidden layers of 100 nodes each and 2 output nodes

After replacing the head layer, training begins by feeding the model with mini batches of data, computing loss using the cross-entropy loss function and using Stochastic Gradient Descent (SGD) as the optimizer to compute the gradient of the loss with respect to all trainable parameters. Both the training phase and validation phase is carried off in each epoch for a total of 10 epochs [13]. At the end, the model with the best accuracy gets saved. Fig. 7 showcases the structure of the model, *M1*, that was trained on task 1. As shown in Fig. 7, *M1* contains 4 layers, and each layer is indicated by its name

and weight and bias values it holds. For example, “classifier.0.weight” is the name of the first layer and “v.0.w” represents the weight values that the first layer contains. Similarly, “v.0.b” represents the bias value that the first layer contains. The rest of the layers are described in the same manner.

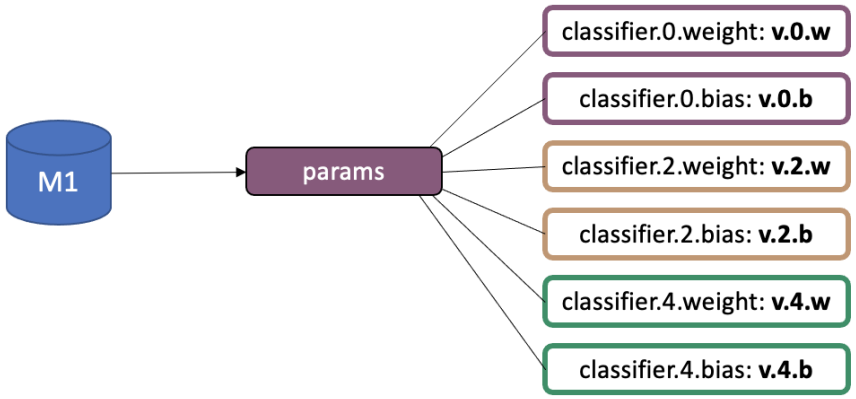


Figure 7: Structure of model, *M1*, trained on task 1

Task 2 is trained differently than task 1. The weights of the model trained on task 1 will be transferred to the model that is going to be used to train task 2. Since we want to preserve the model for task 1 and train the classifier for the new task, we replace the head layer with 2 new units since the outputs of task 2 are [2, 3]. After replacing the head layer, the model is initialized with regularized parameters for each layer by setting a few attributes, that have not been initialized, to 0. The attributes of the regularized parameters are *w, omega, init_val, name*, and *lambda*. The structure of the model, *M2*, trained on task 2 can be seen in Fig. 8. As noted in

Fig. 8, each parameter is regularized. For example, “v.0.w” represents the parameter value of the first layer and the values (i.e. *w: 0, omega:0*) following “v.0.w” represent the regularized parameters of the first layer. The definitions of each of these attributes is found in Table II. The training process for task 2 is like the training process in task 1, except for the notable use of Weighted Stochastic Gradient Descent (SGD). Unlike the traditional SGD, the Weighted SGD method has additional attributes to consider when updating the parameter values during backward propagation.

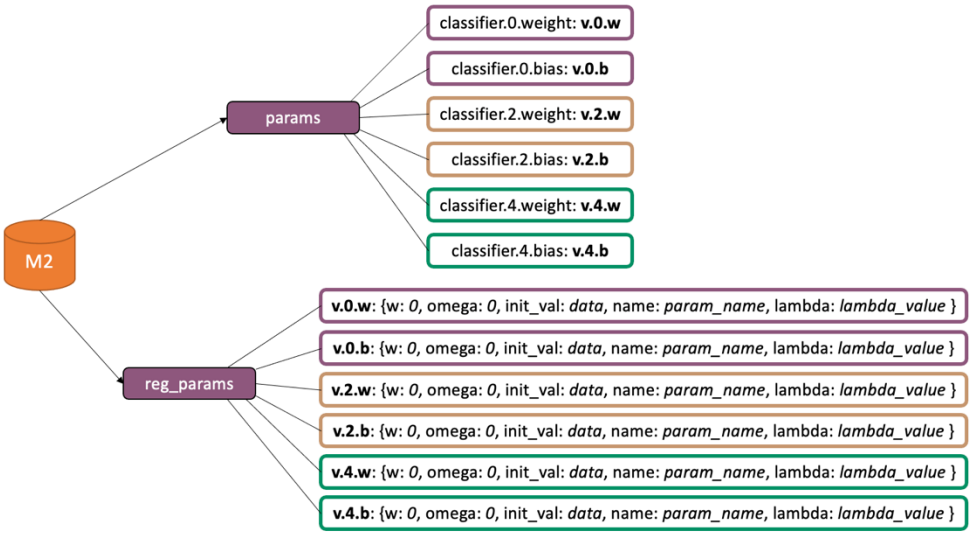


Figure 8: Structure of model, *M2*, trained on task 2

TABLE II: Attributes of Regularized Parameters

Regularized Parameter Attributes	Definition
w	Parameter weights
omega	Importance weights
init_val	Parameter data
name	Parameter name
lambda	Regularization rate

In Weighted SGD, the parameter values in each layer are updated after performing a series of operations. Initially, a weighted difference is computed by subtracting the current weight value of a parameter with the initial value of the parameter. This weighted difference is then multiplied with a constant 2, the *omega* attribute from the regularized parameter, and *lambda*. This product is then added to the parameter's gradient value. If a momentum is defined and the state of the parameter has not been initialized yet, a buffer is created which stores a copy of the parameter's gradient value as part of its initialization. In this case, the value stored in the buffer is the same as the gradient value, so the gradient value remains the same. If a momentum is defined and the state of the parameter has already been initialized, then the value inside the buffer is multiplied with the defined momentum and added with the parameter's gradient value. In both cases, the parameter's data gets updated by adding the gradient data. This process repeats for all the parameters in each layer of the model. Fig. 9 shows how the parameter value gets updated with Weighted SGD for the first layer.

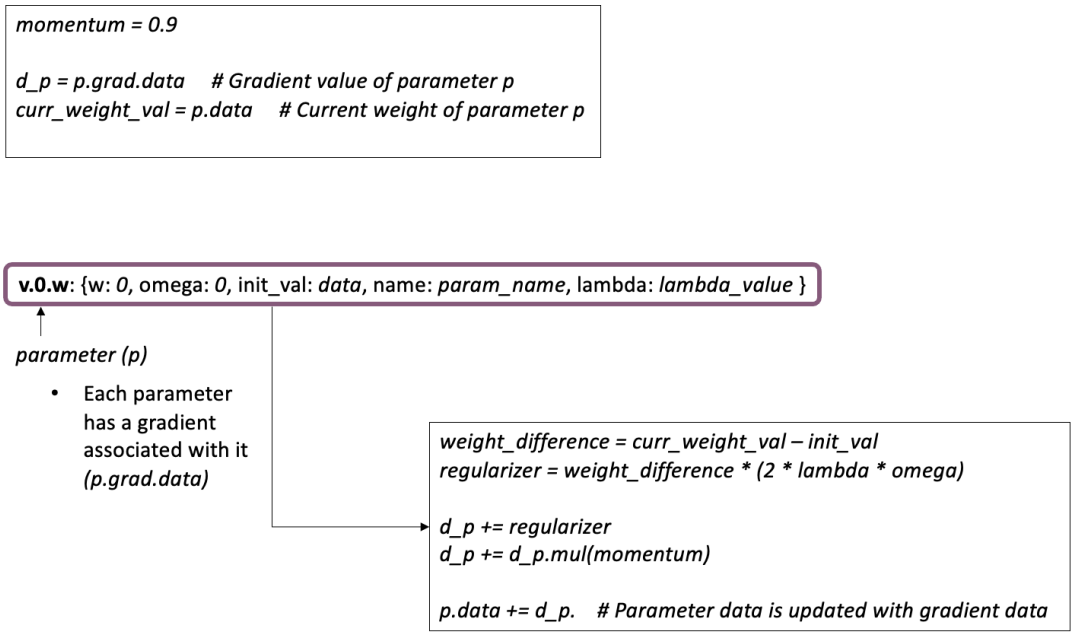


Figure 9: Example of Weighted SGD for first layer for model, *M2*

The training and validation phase is carried off in each epoch for a total of 10 epochs. At the end of each validation phase, accuracy is computed, and the model is saved if it showed improved validation accuracy.

Task 3 is trained like task 2 with a few adjustments. The head layer is replaced with 2 new units since the expected outputs of task 3 are [4, 5]. Recall, regularized parameters were initialized when the model was being trained on task 2. Now, these regularized parameters are updated for the new task. Prior to the update, the regularized parameters for the head layer are removed. For the existing regularized parameters, *omega* is updated to 1 and *init_val* is updated to the data of the parameter. Since the regularized

parameters of the head layer were removed, it's initialized with *omega* to 1 and *init_val* to the data of the parameter. The regularized parameters of the model now only consist of three attributes, *omega*, *init_val*, and *lambda*. Fig. 10 showcases the structure of the model trained on task 3, *M3*. After updating the regularized parameters, the rest of the training and validation process is the same as that of task 2 with Weighted SGD and calculation of average loss and accuracy to find the best model that was trained on task 3. Furthermore, task 4 and 5 in the same way as task 3. After training is completed, five models can be found, each one trained on one task.

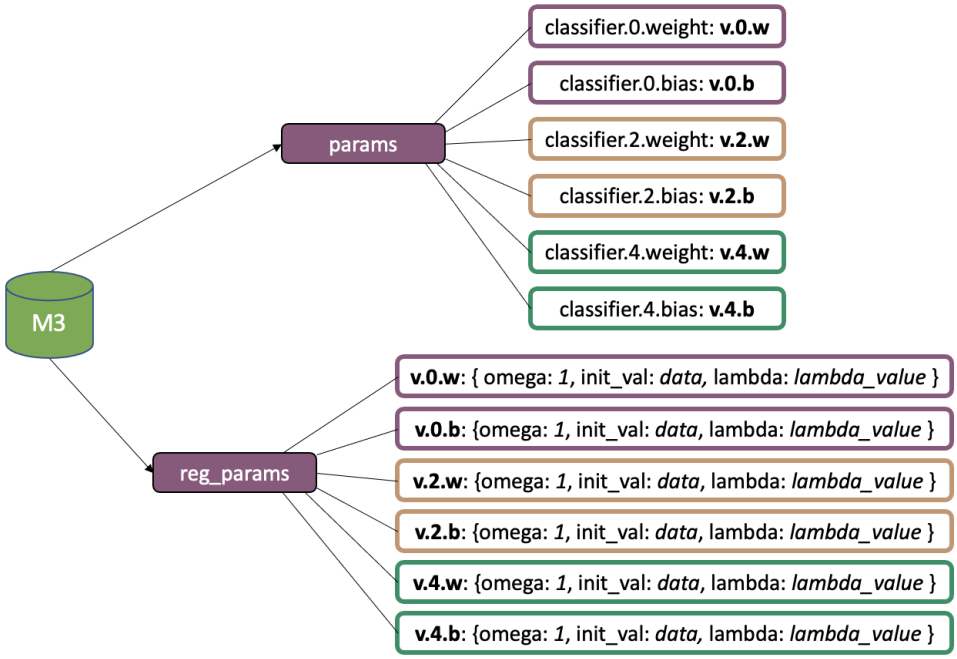


Figure 10: Structure of model trained on task 3, *M3*

b) *IMM Merging Process:* After the server has completed training for each task, each model is used to calculate user priors, or importance weights (IW), from

the user dataset, as part of user adaptation. The IW are then sent to the server to create a merged model for each task with the server trained models and user priors. In this experiment, there are two users. To illustrate the merging process in its entirety, the focus will be on user 1, even though the same method is applied to user 2.

Once the server has finished training on all tasks, a model can be found for each task. In this experiment, there were 5 tasks so 5 trained models can be found, as shown on the right-hand side of Fig. 11. Each of the models is going to be used to calculate IW from the user dataset that theoretically resides on a user device. This user dataset has been divided into appropriate tasks beforehand, as shown on the left-hand side of Fig. 11.

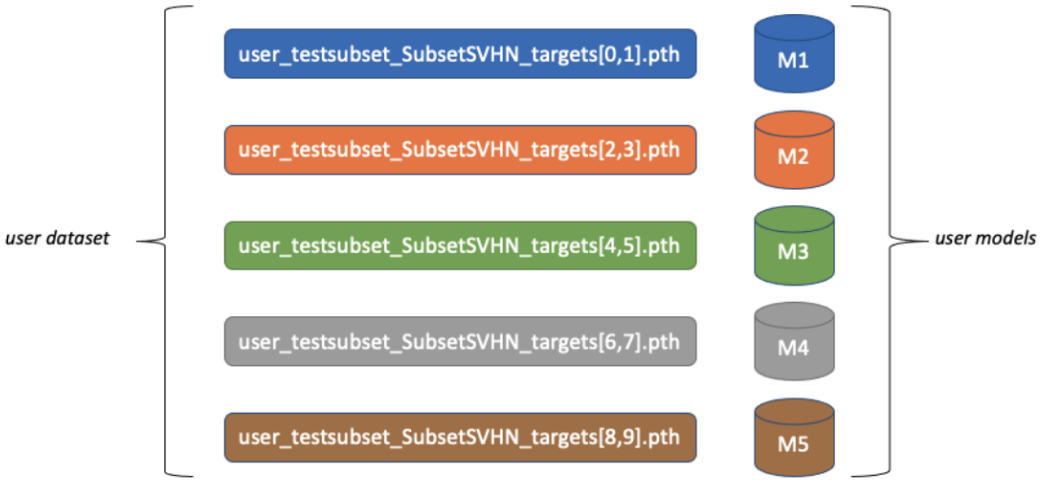


Figure 11: Each user model is used to extract importance weights from user dataset

Prior to calculating the IW and merging models, the regularized parameters for each layer in all models is removed and re-initialized with new attributes and values. The attributes are *omega*, *prev_omega*, and *init_val* which is initialized to 0, the previous omega value and the data of the parameter, respectively. The general structure of each model can be seen in Fig. 12.

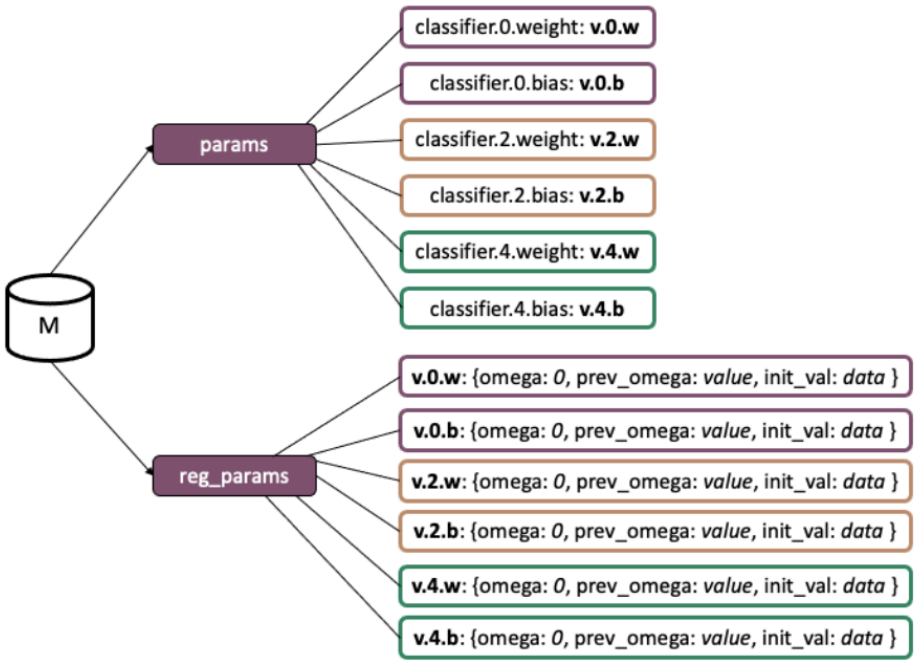


Figure 12: General structure of each model after regularized parameters are removed

A merged model is computed for each task except for the first task by a merging process as defined by IMM. In the IMM merging process, importance weights are computed for each task using L2 norm followed by a calculation of a running total of the importance weights of all tasks. Next, the merging

process begins by first merging task 2 and task 1, followed by merging task 3 with task 2 and task 1, then merging task 4 with task 3, task 2 and task 1, and finally merging task 5 with all previous tasks through a series of operations.

i. Importance Weight Calculation

User 1 data, which consists of images that contain 0 and 1 from the testing SVHN dataset, in addition to images that contain digits 2 through 9, is fed into model *M1* in batches of 20 and the outputs are computed. Once the model outputs are retrieved, a Mean Squared Error (MSE) loss is computed with the model output and accumulated with each batch of data. After each batch of data is processed, the optimization process starts and, in this process, the regularized parameters for each layer in the model are updated through a series of operations. For example, let's say *v.0.w* is the regularized parameter that represents weights in the first layer of the model. This parameter has attributes *omega*, *prev_omega*, and *init_val* which are initialized to 0, some value and data of the parameter respectively. The *omega* attribute is multiplied with the previous size of the data, added with the gradient data of the parameter, and divided by the current size of the data set in this sequence. The *omega* value gets updated for *v.0.w*. An illustration of this can be found in Fig. 13. This same process is applied to all the parameters until all the user data has been processed. At the end, L2 norm is computed by dividing the accumulated

data with total data and *M1* is returned with updated regularized parameters as seen in Fig. 14.

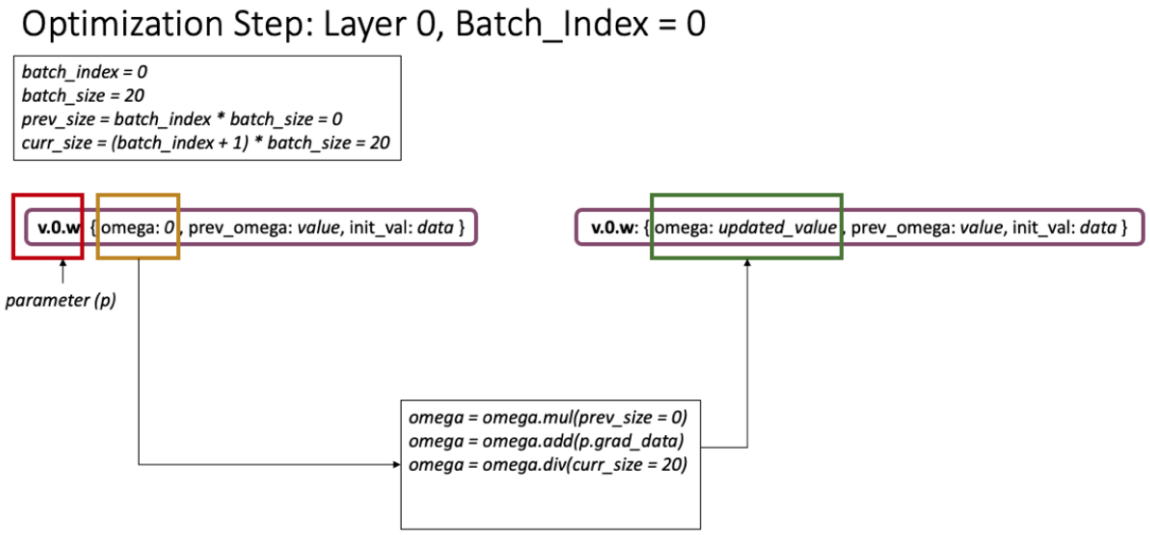


Figure 13: Optimization process of updating regularized parameters

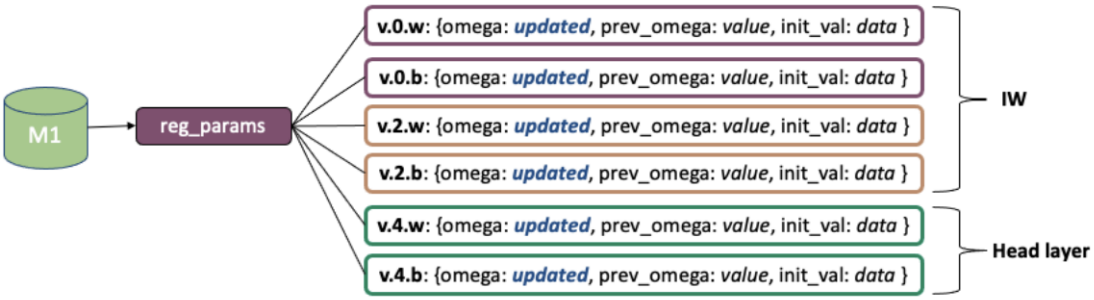


Figure 14: Model, *M1*, with updated regularized parameters

This process is repeated for all tasks and corresponding models. It's important to note that the importance weights are the updated *omega* values and importance weights of the head layer are not considered during the merging process as indicated in Fig. 14. This IW calculation is

done for each task and an illustration of the importance weights for each task can be seen in Fig. 15.

Figure 15: Importance weights of each of the 5 tasks

ii. Running Sum of Importance Weights Calculation

The running sum of importance weights is initialized with the IW of task 1. Subsequently, after the calculation of the IW of task 2, the running sum of importance weights is accumulated with the IW of task 2 as shown in Fig. 16. This process continues until all the importance weights for all tasks are considered, as seen in Fig. 17.

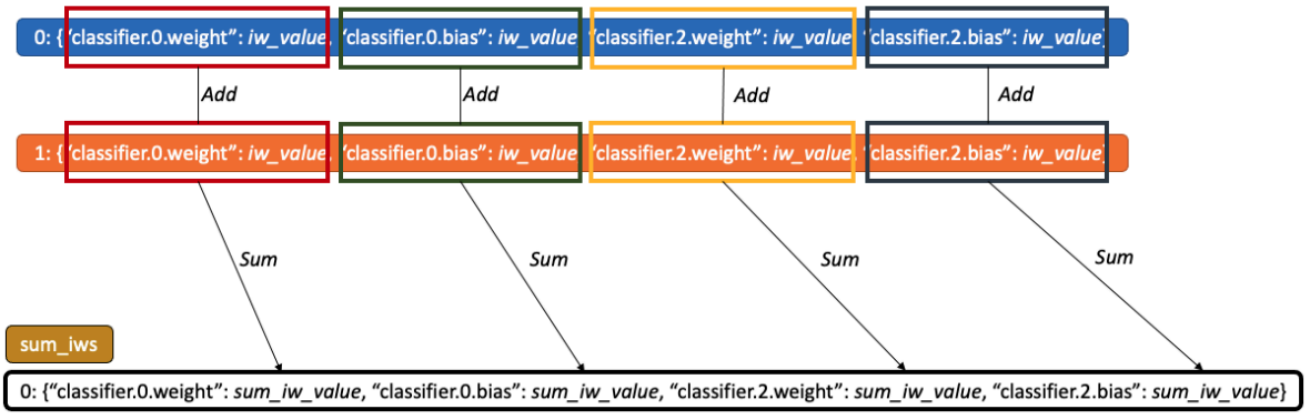


Figure 16: Summation of importance weights of task 1 and task 2

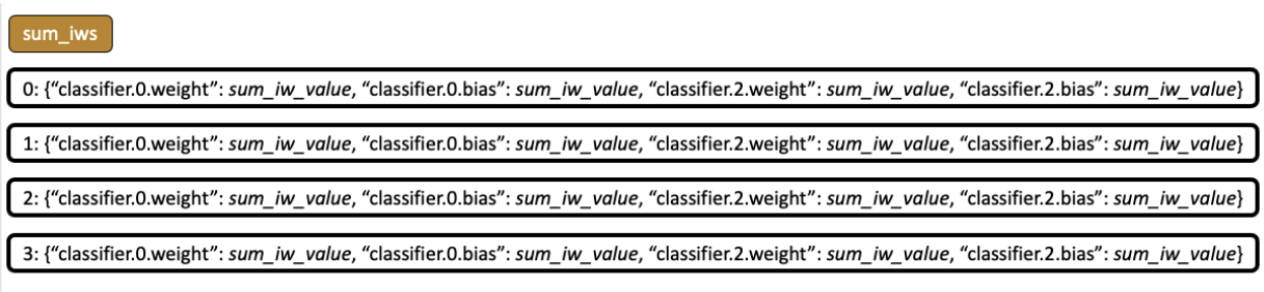


Figure 17: Summation of importance weights for all tasks, excluding task 1

iii. Merging

Recall that task 1 does not have a merged model associated with it because it's the first task and there is no previous task for it to merge with. Initially, a merged model is created for task 2 by merging it with task 1 through a series of steps.

1. Calculate the weightage (*importance*) of the parameter values in each layer of the task 2 model by dividing the IW of task 1 with the running sum value that is averaged over task 1 and 2.
2. Multiply the weightage with the parameter values of the corresponding layer
3. Create a running sum of product
4. Repeat the process until there are no more tasks to merge
5. Update the parameter value with the mean that is computed by dividing the running sum with the total number of tasks that were to be merged

This same set of steps are followed by merging task 3 with task 2 and task 1, merging task 4 with task 3, task 2, and task 1, and lastly merging task 5 with all previous tasks. At the end of the merging process, there should be four merged models for all tasks except for the first task.

2) MIT Indoor Scenes Experiments: The MIT Indoor Scenes dataset was used in the last two experiments. Both these experiments incrementally train a VGG11 (pretrained on ImageNet) model on a series of tasks defined by the MIT Indoor Scenes dataset. This dataset consists of 5 super categories and each super category has subcategories associated with it. The 5 super categories are *store, home, public spaces, leisure and working place*. Each of these super categories, aside from working place, is defined as a task. A clear picture of the defined tasks can be found in Table III. The MIT Indoor Scenes dataset contains 67 indoor categories and a total of 15620 images. However, in both experiments, only 5360 images were used for training and only 1,340 images were used for testing.

TABLE III: Tasks of MIT Indoor Scenes

TASKS	SUBSET OF INDOOR SCENES
1	home
2	leisure
3	public
4	store

a) User Transform: The first experiment on the MIT Indoor Scenes dataset considered a total of 10 users and each user is allocated an equal amount of data from all categories in all tasks from the testing set of images. For example, user 1 has been allocated 635 evaluation samples and 639 importance weight samples from task 2. User 2 also has the same number of evaluation and importance weight samples from task 2. In other words, no user has preferences over the categories and importance weights are calculated across all categories. In addition to this change, a transform is applied to the training and validation dataset on the server and a different transform is applied to each user dataset as an additional privacy measure.

i. Server Training of Models for all Tasks

All 4 tasks are trained incrementally on the server using IMM just like in the numbers experiment. This experiment starts with a VGG11 model that was pretrained on ImageNet. Prior to training task 1, the classifier layer of the VGG11 model is replaced with 14 new output units because task 1 contains 14 subcategories. After replacing the head layer, training begins by feeding the model with data of mini batches of size 30, computing loss using the cross-entropy loss function and using Stochastic Gradient Descent (SGD) as the optimizer to compute the gradient of the loss with respect to all trainable parameters. Both the training phase and validation phase is carried off in each epoch. There are a total of 49

epochs defined, however, it doesn't run through all the epochs if a best model has already been found. At the end of the validation phase in each epoch, the model with the highest accuracy gets saved for task 1.

The model that will be used to train task 2 is the model that was trained on task 1 through weight transfer. Since we want to preserve the model for task 1, we replace the head layer with 11 new units since there are 11 subcategories in task 2. After replacing the head layer, the model is initialized with regularized parameters for each layer by setting a few attributes, that have not been initialized, to 0. The attributes of the regularized parameters are *w, omega, init_val, name*, and *lambda*. The definitions of each of these attributes is found in Table II. The training process of task 2 is like the training process of task 1. Just as in the numbers experiment, the Weighted Stochastic Gradient Descent (SGD) is used as the optimizer. At the end of the training and validation phase in each epoch, accuracy is compared with the best accuracy and the best accuracy value gets updated to the current accuracy for the next epoch and the best model gets saved for task 2.

Task 3 is trained like task 2 with a few adjustments. The head layer is replaced with 14 new units since the outputs of task 3 are 14 subcategories. Recall, regularized parameters were initialized when the model was being trained on task 2. Now, these regularized parameters

are updated for the new task. Prior to the update, the regularized parameters for the head layer are removed. For the existing regularized parameters, *omega* is updated to 1 and *init_val* is updated to the data of the parameter. Since the regularized parameters of the head layer were removed, its initialized with *omega* to 1 and *init_val* to the data of the parameter. The regularized parameters of the model now only consist of three attributes, *omega, init_val*, and *lambda*. After updating the regularized parameters, the rest of the training and validation process is the same as that of task 2 with Weighted SGD and calculation of average loss and accuracy to find the best model that was trained on task 3. Furthermore, task 4 is trained in the same way as task 3. After training is completed, four models can be found, each one trained on one task.

ii. IMM Merging Process

After the server has completed training for each task, each model is used to calculate user priors, or importance weights (IW), from the user dataset, as part of user adaptation. In this experiment, there are ten users. Recall that each of these 10 users do not have a preference over the categories in each task. Rather, the user dataset is composed of images from all tasks, and it's divided into an evaluation dataset and an importance weight dataset. As the name indicates the importance weights are calculated from the importance weight dataset. To illustrate the

merging process in its entirety, the focus will be on user 1, even though the same method is applied to the rest of the users.

Once the server has finished training on all tasks, a model can be found for each task. In this experiment, there were 4 tasks so 4 trained models can be found. Each of the models is going to be used to calculate IW from the user dataset, which has been divided into appropriate tasks beforehand. Prior to calculating the IW and merging models, the regularized parameters for each layer in all models is removed and re-initialized with new attributes and values. The attributes are *omega, prev_omega*, and *init_val* which is initialized to 0, previous omega value and the data of the parameter, respectively. *Please refer to the importance weight section in the numbers experiment to see how importance weights are calculated.*

A merged model is computed for each task except for the first task by a merging process as defined by IMM. In the IMM merging process, importance weights are computed for each task using L2 norm followed by a calculation of a running total of the importance weights of all tasks. Next, the merging process begins by first merging task 2 and task 1, followed by merging of task 3 with task 2 and task 1, then merging task 4 with task 3, task 2 and task 1. *Please refer to the merging process section*

described in the numbers experiment because the same process is applied for this experiment.

b) Category Prior: In contrast to the first experiment on the MIT Indoor Scenes dataset, the second experiment assigns preference of 3 categories to each of the 5 users that this experiment considers. Users are allocated data based on the categories they prefer from all tasks. For example, user 1 has been allocated 132 evaluation samples and 133 importance weight samples with preferences for *winecellar, dining room* and *corridor* from task 2. User 2 has 131 evaluation samples and 133 importance weight samples with preferences for *dining room, corridor,* and *winecellar.* Also, unlike the first experiment, there is no transform applied to the training and validation dataset on the server nor on the user dataset. *The training, merging, and testing processes for this experiment is the same as the user transform experiment so, please refer to that section for concepts.*

VI. PRELIMINARY RESEARCH

A. Analysis of DUA Framework

A few questions arise upon closer inspection of the three experiments that were executed on the DUA framework. In a real-world setting, there is no guarantee that the dataset contained in the user device is like the dataset contained in the server. However, in all three experiments, the same datasets were used for both the server and user devices. For example, in the numbers experiment, the MNIST and SVHN datasets

were used for both training on the server and data on user devices. Upon further observation, in all three experiments, the dataset in the user device is divided into a series of tasks just as the server. In other words, the number of tasks in the user device is the same as the number of tasks in the server. In addition to the fact that there is no guarantee that the user data will be like server data, there is no guarantee that user data will be divided into a series of tasks since user data is continuously evolving. These observations suggest that the DUA framework is built upon a series of assumptions. A question that arises then is that, how is the performance of server trained models and models delivered to the user on unseen data? Therefore, it's important to investigate the robustness of these models on unseen data.

1) *Implementation Plan to Check Robustness:*

    a) *Numbers Experiment:*

        In this experiment, as defined in the paper, each merged model was tested on all task data. For example, the merged model in task 2 was tested on task 1 data and evaluated by replacing its head layer with that of the model trained on task 1. This testing and evaluation process was done for all merged models. It's important to mention that the testing data that was used to evaluate the performance of all merged models on all tasks was the same as the user dataset that was initially used to calculate importance weights for the merging process. Hence, it's essential to check the robustness of these merged models by evaluating it on unseen data and we are using .

1. Pre-process the EMNIST digits dataset by applying a series of transforms
2. Divide the EMNIST digits dataset into the same 5 tasks as defined in the server by both MNIST and SVHN datasets
3. Feed the unseen EMNIST digits task data into the server trained models and evaluate how each model performs on unseen data
4. Repeat the previous step for the merged models for user 1 and evaluate how each merged model performs on unseen data
5. Replace the head layer of each unmerged model with the head layer of all models
6. Feed the unseen EMNIST digits data into each corresponding newly configured model and evaluate how each unmerged model performs on unseen data
7. Repeat the previous two steps for merged models and evaluate how each merged model performs on unseen data

b) MIT Indoor Scenes Experiment: There were two experiments that used the MIT indoor scenes dataset, user transform and category prior. However, the robustness check will only be performed on the category prior experiment for the purpose of illustrating how robustness is done.

In this experiment, each merged model was tested on all task data. For example, the task 2 merged model was tested on task 1 data and evaluated by

replacing its head layer with that of the model trained on task 1. This testing and evaluation process was done for all merged models. It's important to mention that the testing data that was used to evaluate the performance of all merged models on all tasks was the evaluation dataset that was created from the MIT indoor scenes dataset. Hence, it's essential to check the robustness of these models by evaluating it on unseen data.

1. Gather non-commercial images from all categories in each task from Google using the google-images-download tool
2. Divide the custom scenes dataset into tasks like the tasks in the experiment.
3. Feed the unseen scenes task data into the server trained models and evaluate how each model performs on unseen data
4. Repeat the previous step for the merged models for user 1 and evaluate how each merged model performs on unseen data
5. Implement and train a centralized model with user data and compare its performance with that of the DUA framework
6. Implement FedAvg and train the framework using MIT indoor scenes dataset to compare its performance with that of the DUA framework
7. Repeat step 6 with FedProx framework

2) Evaluation Results:

a) Robustness Check of Numbers Experiment:

i. Unmerged Models

The first robustness check that was done was to see how well all server trained models (unmerged models) performed on unseen EMNIST task data. Recall that there is a model for every task after the server is done training. Appropriate task data was fed into corresponding task models in batches of 50 and an average accuracy was computed after all the data had been processed. Fig. 18 displays the confusion matrices for all unmerged models. The first confusion matrix that is shown in Fig. 18 is that of task 1 unmerged model and it shows that 3969 samples out of 4000 were correctly classified as 0 and 3931 samples out of 4000 were correctly classified as 1. It also shows that 31 samples out of 4000 were misclassified as 1 when the expected value was supposed to be 0 and 69 out of 4000 samples were misclassified as 0 when the expected value was supposed to be 1. All the confusion matrices in Fig. 18 indicate the number of correct classifications and misclassifications of each task.

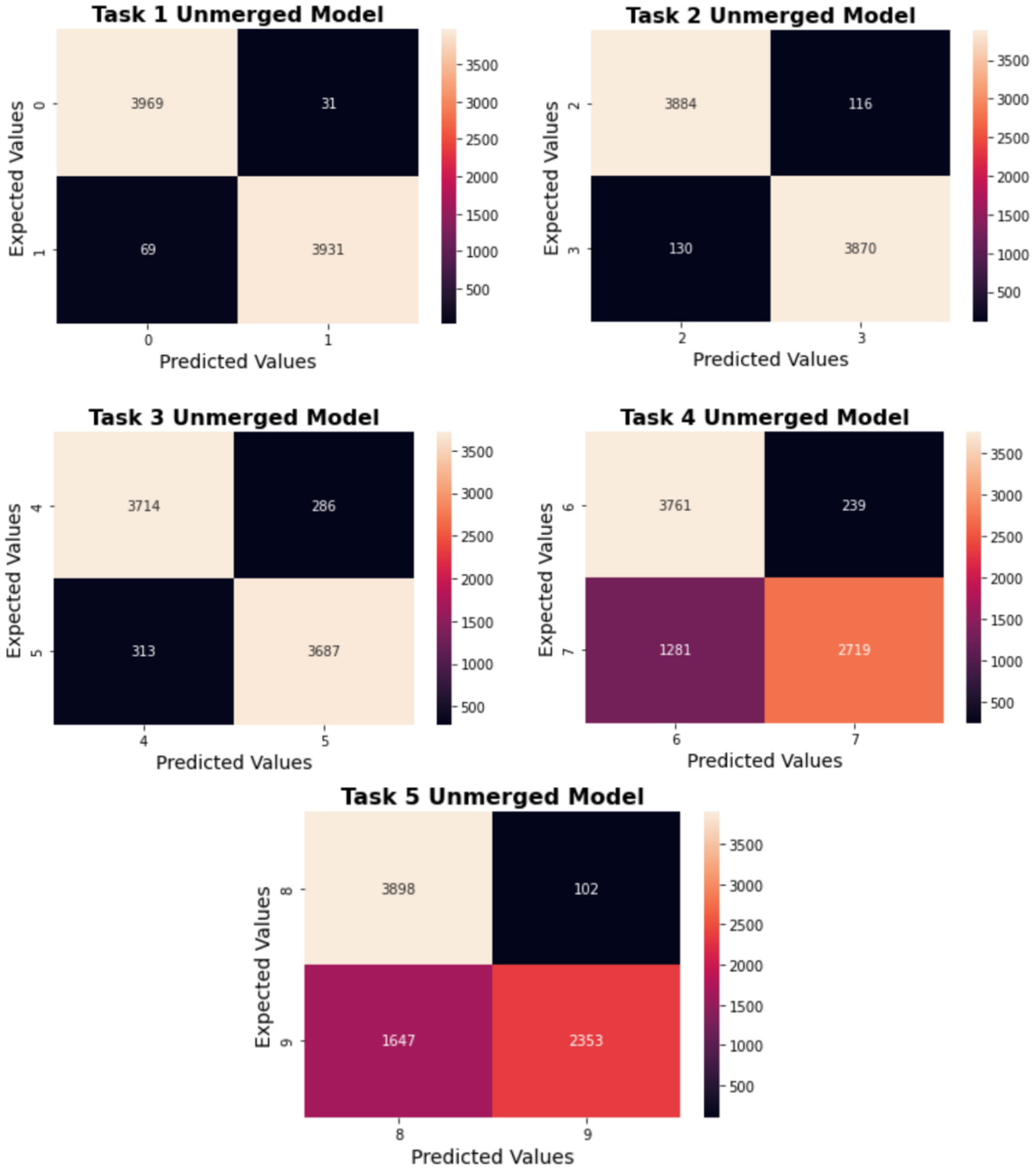


Figure 18: Confusion Matrices for all 5 unmerged models

Fig. 19 provides the average accuracy values for task models based on the confusion matrices that were received on the EMNIST dataset. Based on the results, the average accuracy was high for task 1, 2 and 3 models and somewhat high for tasks 4 and 5. Perhaps, one reason why the accuracy wasn't high for tasks 4 and 5 when compared with the accuracies for tasks 1, 2 and 3 is due to the updated regularized parameters as part of the training process. Nonetheless, the task models do well in predicting the tasks they were trained for on unseen task data.

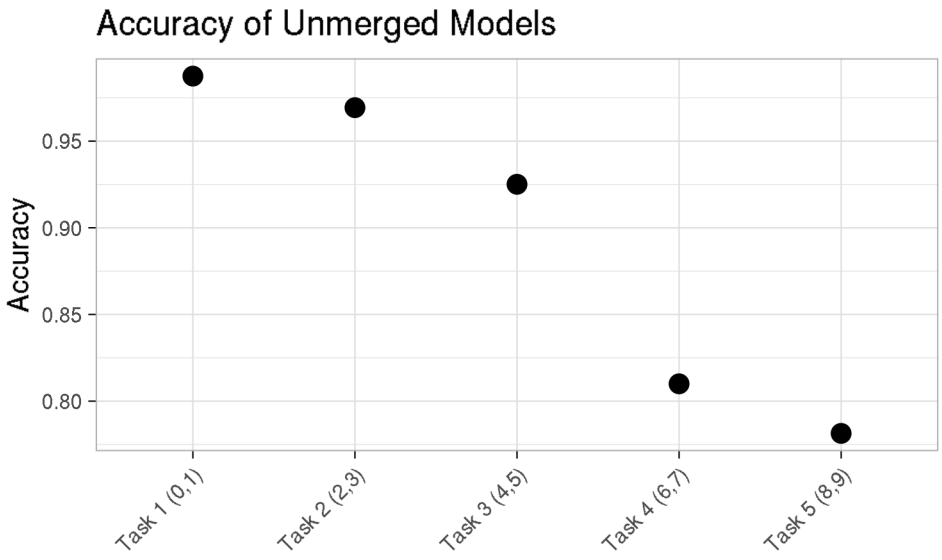


Figure 19: Plot of average accuracies for all 5 tasks

ii. Merged Models

The second robustness check was to verify that each merged model performed correctly on its corresponding task. Each parameter in a merged model contains the mean parameter value of its previous tasks. For example,

the weights in the merged model of task 2 is the average of weights in both task 1 model and task 2 model. Like how the first robustness check was done, appropriate task data was fed into corresponding task merged models in batches of 50 and an average accuracy was computed after all the data had been processed. Fig. 20 displays the confusion matrices for all merged models.

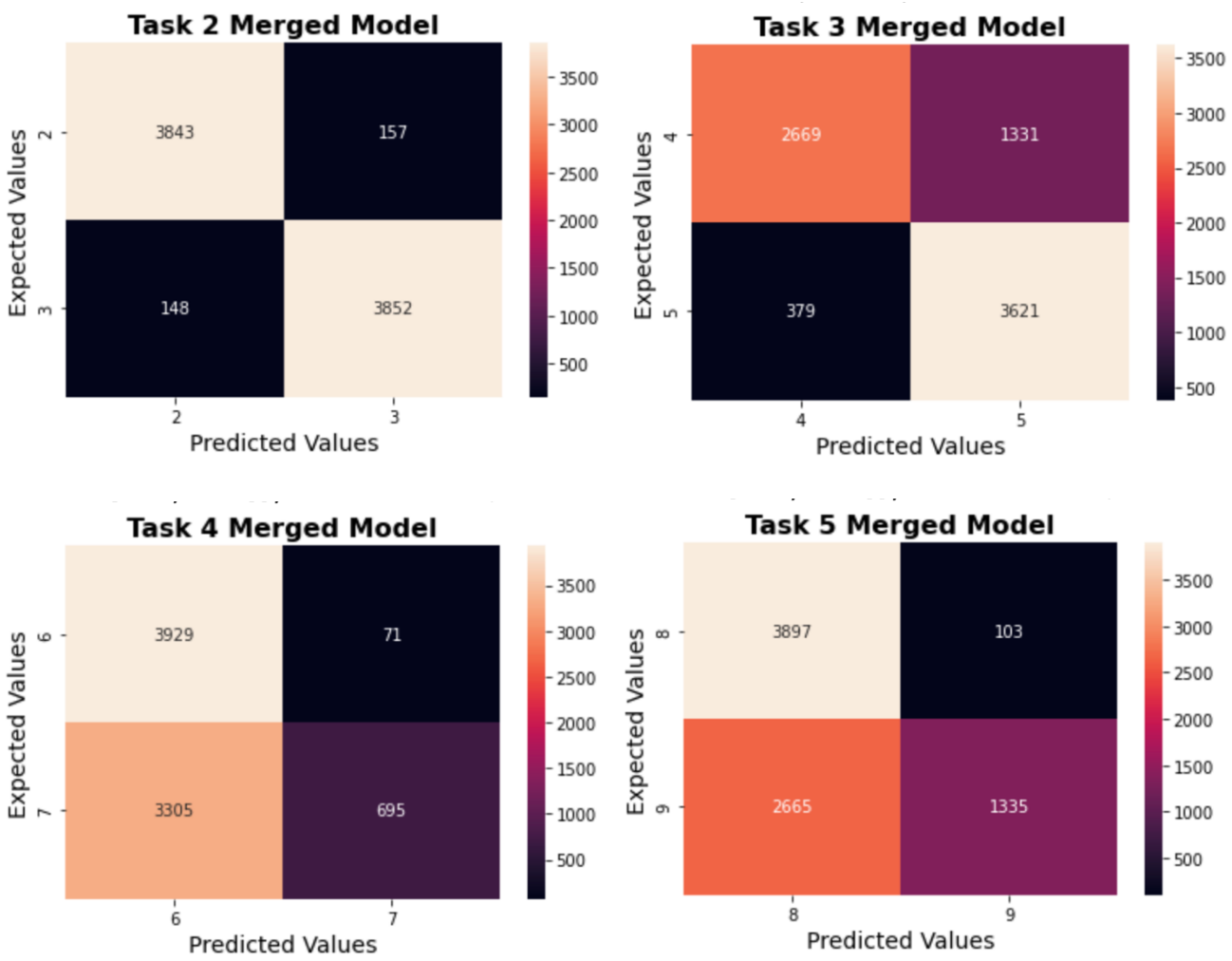


Figure 20: Confusion Matrices for all 4 merged models

Fig. 21 provides the average accuracy values for all merged models that were received on the EMNIST dataset.

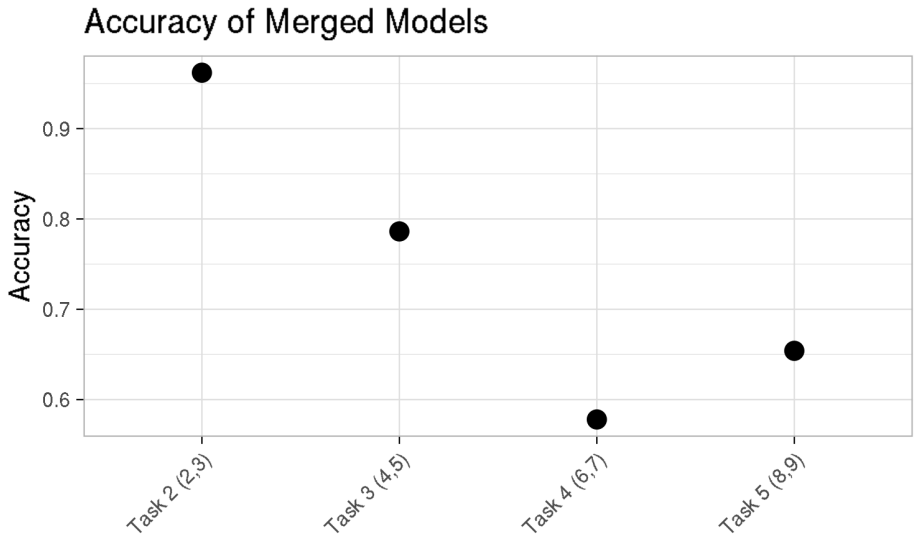


Figure 21: Plot of average accuracies for all 4 merged models

Task 1 does not have a merged model. Based on the results in Fig. 21, the accuracy on the merged model for task 2 is slightly less than the accuracy on the unmerged model for task 2 because the weights of the merged model are averaged over two tasks, task 1 and 2. The accuracies for tasks 3, 4 and 5 are substantially low and this could be because weights are averaged over all previous tasks. This suggests that some of the information that is learned is lost when models are merged.

iii. Unmerged Models with Classifier Layers of all Models

Compared to the previous two evaluation experiments, the third robustness check is done differently. In this evaluation study, the classifier layer

of each model will be replaced with classifier layers of all models. For example, the classifier layer of task 1 unmerged model is originally composed of 2 nodes as in Fig. 6. After replacing this layer with classifier layers from all unmerged models, the classifier layer of task 1 will be composed of 10 nodes. Fig. 22 showcases the architecture of the new model configuration.

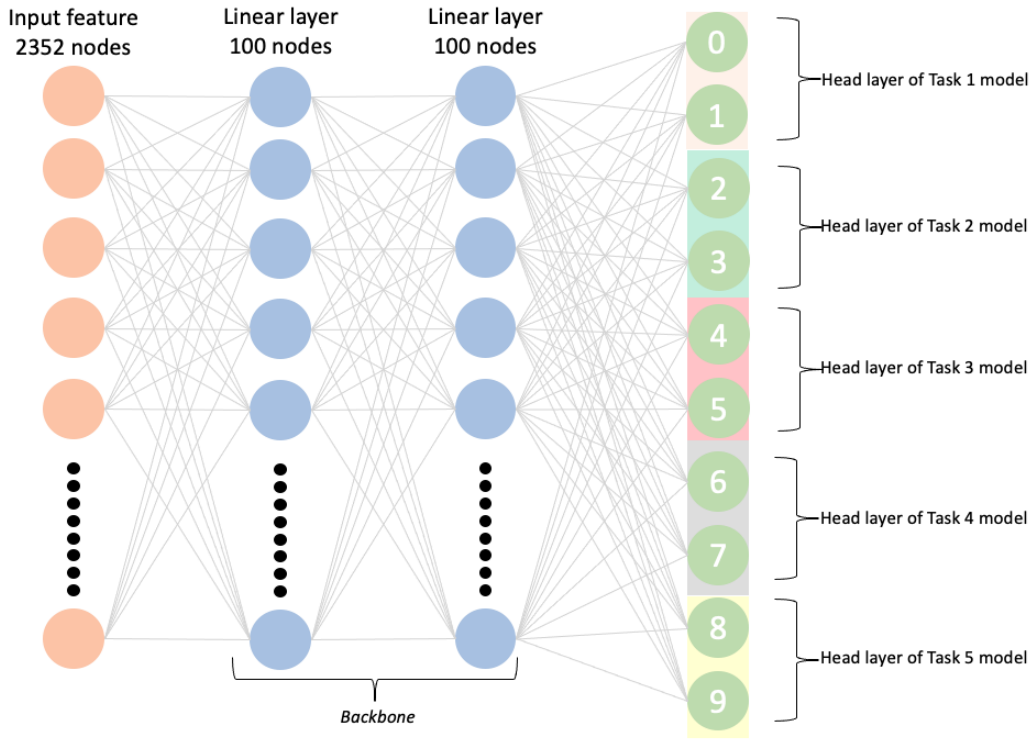


Figure 22: Model configuration with classifier layers of all models

The new model configuration in Fig. 22 is applied to all the unmerged models by replacing the backbone architecture with each unmerged model. For example, when the backbone architecture is the unmerged model for task 1, the new model configuration can be seen in Fig. 23.

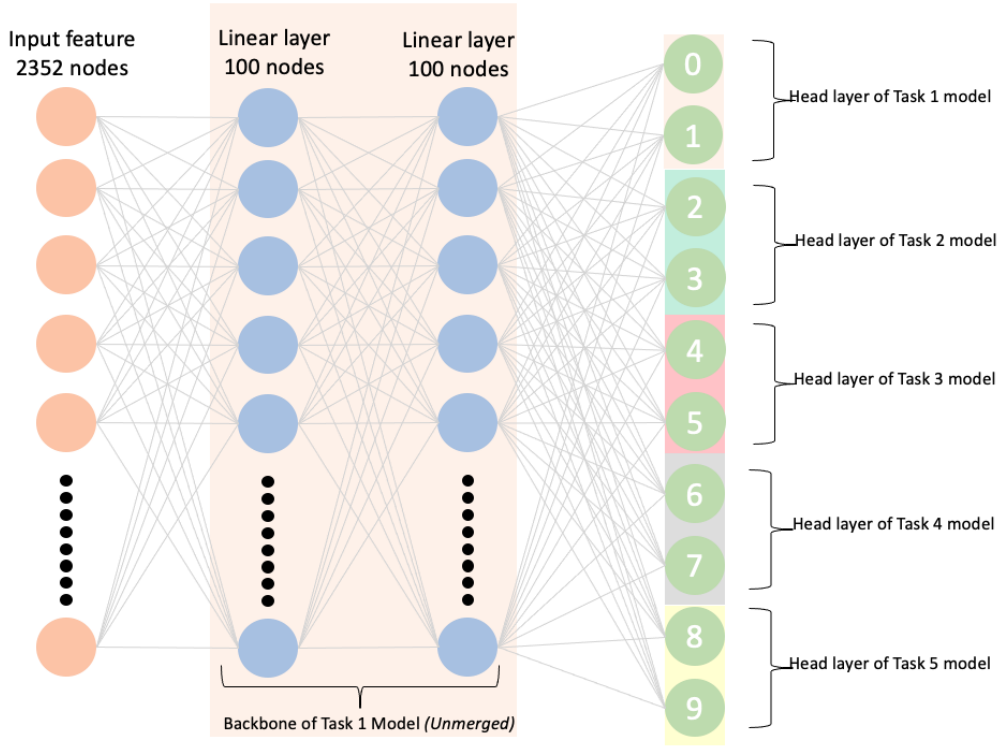


Figure 23: Task 1 unmerged model as backbone with classifier layers of all models

Fig. 24 and Table IV contain the confusion matrix and performance metrics for task 1 unmerged model, respectively. Fig. 25 and Table V contain the confusion matrix and performance metrics for task 2 unmerged model, respectively. Fig. 26 and Table VI contain the confusion matrix and performance metrics for task 3 unmerged model, respectively. Fig. 27 and Table VII contain the confusion matrix and performance metrics for task 4 unmerged model, respectively. Fig. 28 and Table VIII contain the confusion matrix and performance metrics for task 5 unmerged model, respectively.

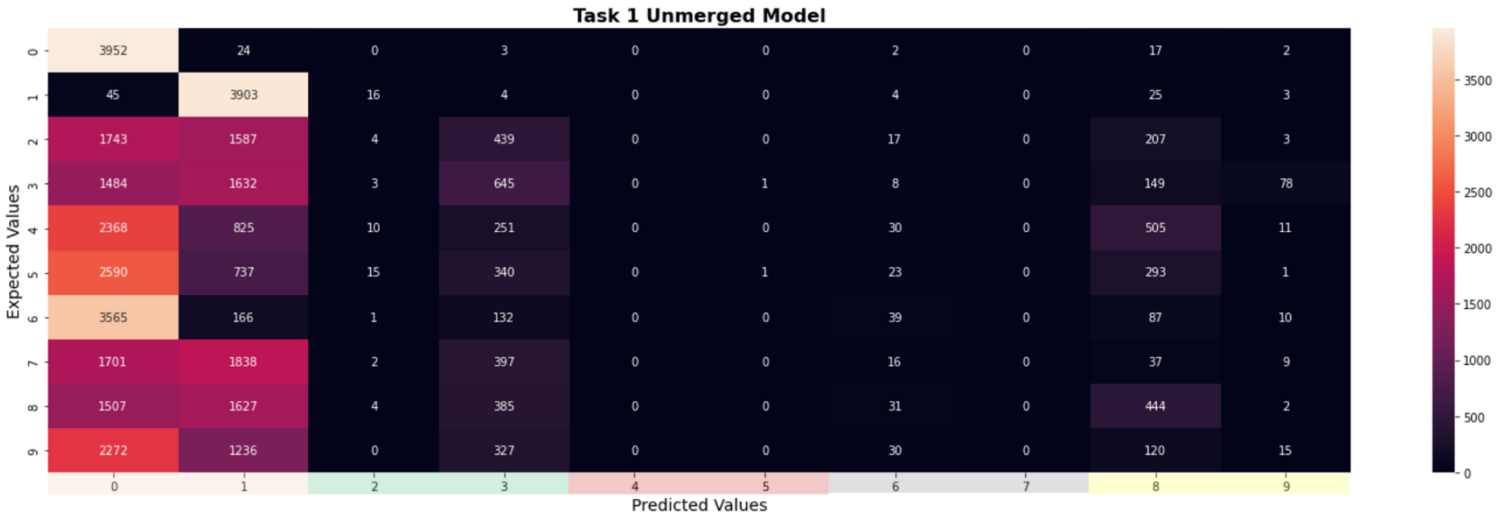


Figure 24: Confusion Matrix for Task 1 Unmerged Model on entire EMNIST dataset

TABLE IV: Performance Metrics of Task 1 Unmerged Model for EMNIST dataset

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	0	0.98800	0.186178	0.98800	0.313315	3952	17275	18725	48	unmerged_T1_backbone_all_heads
1	1	0.97575	0.287514	0.97575	0.444154	3903	9672	26328	97	unmerged_T1_backbone_all_heads
2	2	0.00100	0.072727	0.00100	0.001973	4	51	35949	3996	unmerged_T1_backbone_all_heads
3	3	0.16125	0.220664	0.16125	0.186335	645	2278	33722	3355	unmerged_T1_backbone_all_heads
4	4	0.00000	0.000000	0.00000	0.000000	0	0	36000	4000	unmerged_T1_backbone_all_heads
5	5	0.00025	0.500000	0.00025	0.000500	1	1	35999	3999	unmerged_T1_backbone_all_heads
6	6	0.00975	0.195000	0.00975	0.018571	39	161	35839	3961	unmerged_T1_backbone_all_heads
7	7	0.00000	0.000000	0.00000	0.000000	0	0	36000	4000	unmerged_T1_backbone_all_heads
8	8	0.11100	0.235669	0.11100	0.150918	444	1440	34560	3556	unmerged_T1_backbone_all_heads
9	9	0.00375	0.111940	0.00375	0.007257	15	119	35881	3985	unmerged_T1_backbone_all_heads

Based on the confusion matrix and performance metrics for task 1 unmerged model, there was a high accuracy for predicting 0 and 1, which is expected since this model was trained for classifying 0 and 1.

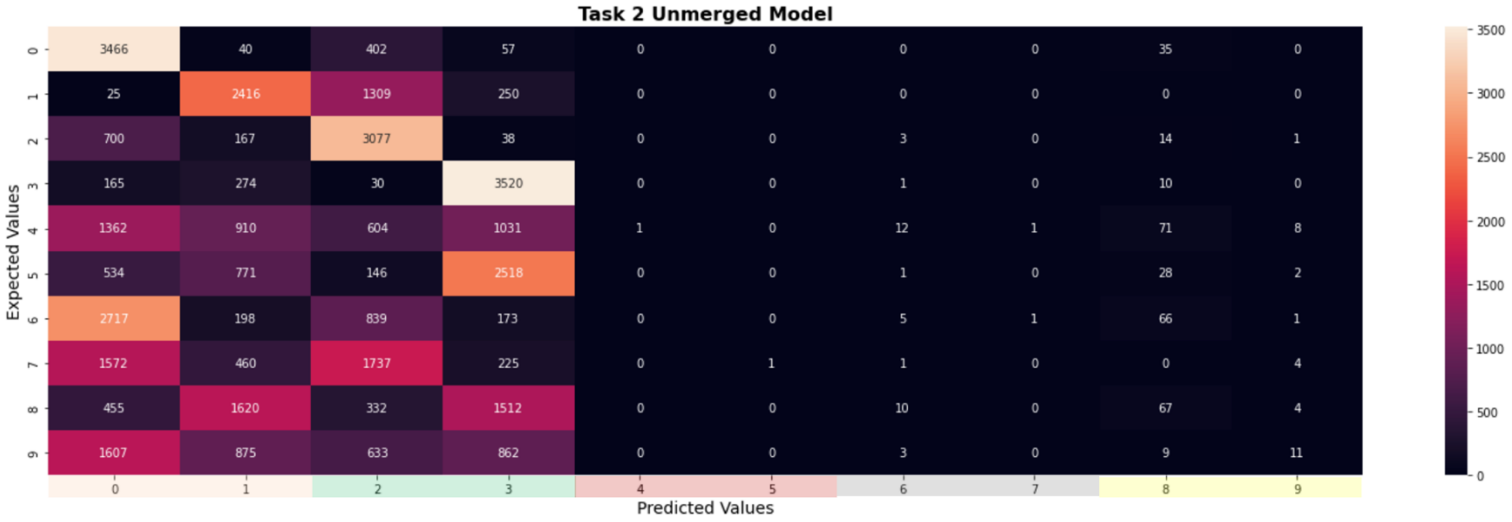


Figure 25: Confusion Matrix for Task 2 Unmerged Model on entire EMNIST dataset

TABLE V: Performance Metrics of Task 2 Unmerged Model for EMNIST dataset

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	0	0.86650	0.275014	0.86650	0.417515	3466	9137	26863	534	unmerged_T2_backbone_all_heads
1	1	0.60400	0.312508	0.60400	0.411900	2416	5315	30685	1584	unmerged_T2_backbone_all_heads
2	2	0.76925	0.337798	0.76925	0.469448	3077	6032	29968	923	unmerged_T2_backbone_all_heads
3	3	0.88000	0.345572	0.88000	0.496264	3520	6666	29334	480	unmerged_T2_backbone_all_heads
4	4	0.00025	1.000000	0.00025	0.000500	1	0	36000	3999	unmerged_T2_backbone_all_heads
5	5	0.00000	0.000000	0.00000	0.000000	0	1	35999	4000	unmerged_T2_backbone_all_heads
6	6	0.00125	0.138889	0.00125	0.002478	5	31	35969	3995	unmerged_T2_backbone_all_heads
7	7	0.00000	0.000000	0.00000	0.000000	0	2	35998	4000	unmerged_T2_backbone_all_heads
8	8	0.01675	0.223333	0.01675	0.031163	67	233	35767	3933	unmerged_T2_backbone_all_heads
9	9	0.00275	0.354839	0.00275	0.005458	11	20	35980	3989	unmerged_T2_backbone_all_heads

Based on the confusion matrix and performance metrics of task 2 unmerged model, 3077 samples out of 4000 were correctly classified as 2 and 3520 samples out of 4000 were correctly classified as 3. The results also show that 3466 samples out of 4000 were classified as 0 and 2416 were classified as 1. Even though the accuracy of identifying 0 and 1 was low compared to the accuracy of identifying 0 and 1 when the backbone was task 1 unmerged model, this shows that some of the learned features haven't been forgotten since the models have been trained sequentially in the DUA framework.

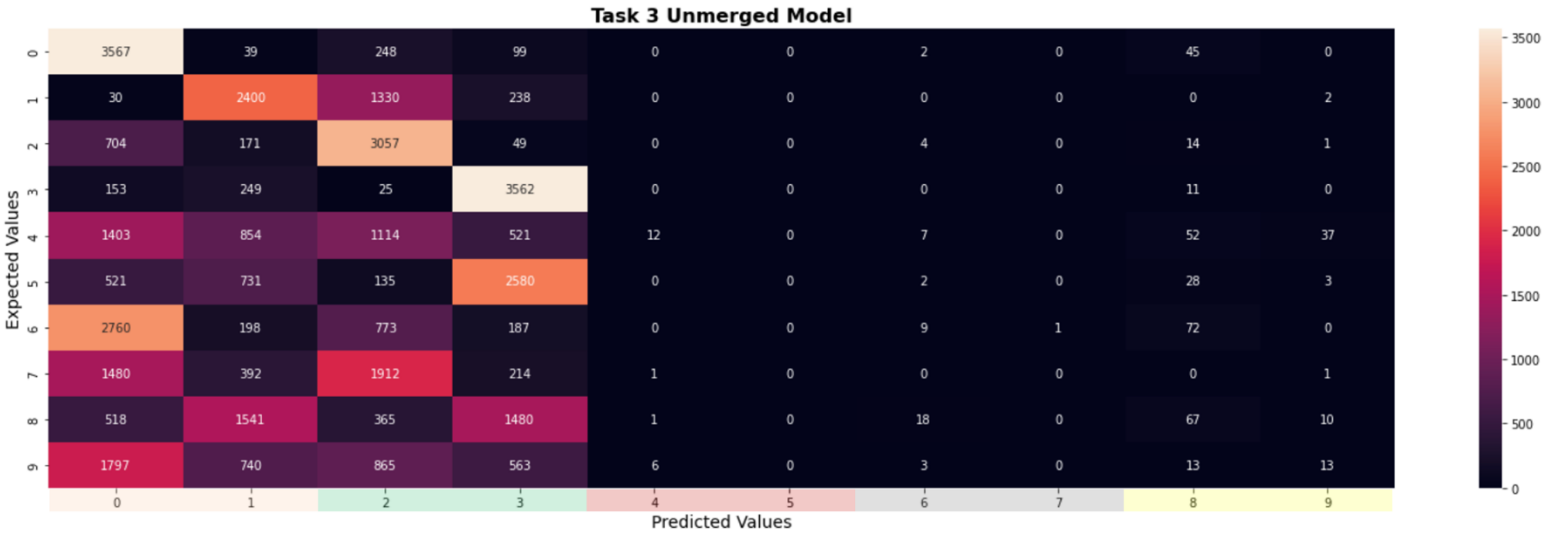


Figure 26: Confusion Matrix for Task 3 Unmerged Model on entire EMNIST dataset

TABLE VI: Performance Metrics of Task 3 Unmerged Model for EMNIST dataset

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	0	0.89175	0.275806	0.89175	0.421308	3567	9366	26634	433	unmerged_T3_backbone_all_heads
1	1	0.60000	0.328093	0.60000	0.424216	2400	4915	31085	1600	unmerged_T3_backbone_all_heads
2	2	0.76425	0.311177	0.76425	0.442274	3057	6767	29233	943	unmerged_T3_backbone_all_heads
3	3	0.89050	0.375224	0.89050	0.527977	3562	5931	30069	438	unmerged_T3_backbone_all_heads
4	4	0.00300	0.600000	0.00300	0.005970	12	8	35992	3988	unmerged_T3_backbone_all_heads
5	5	0.00000	0.000000	0.00000	0.000000	0	0	36000	4000	unmerged_T3_backbone_all_heads
6	6	0.00225	0.200000	0.00225	0.004450	9	36	35964	3991	unmerged_T3_backbone_all_heads
7	7	0.00000	0.000000	0.00000	0.000000	0	1	35999	4000	unmerged_T3_backbone_all_heads
8	8	0.01675	0.221854	0.01675	0.031148	67	235	35765	3933	unmerged_T3_backbone_all_heads
9	9	0.00325	0.194030	0.00325	0.006393	13	54	35946	3987	unmerged_T3_backbone_all_heads

Based on the confusion matrix and performance metrics of task 3 unmerged model, only 12 out of 4000 samples were correctly classified as 4 and no samples were classified as 5 even though the dataset contained 4000 images that contained the digit 5. Another observation is that 3567 samples out of 4000 have been classified as 0 and 3562 samples out of 4000 have been classified as 3. Task 3 model was trained on classifying 4 and 5, however, it performed poorly on classifying 4 and 5 correctly compared to its classification of 0 and 3. This indicates that the output scores for classes 0 and 3 may be higher than the output scores for 4 and 5 and the model isn't well calibrated to identify 4 and 5.

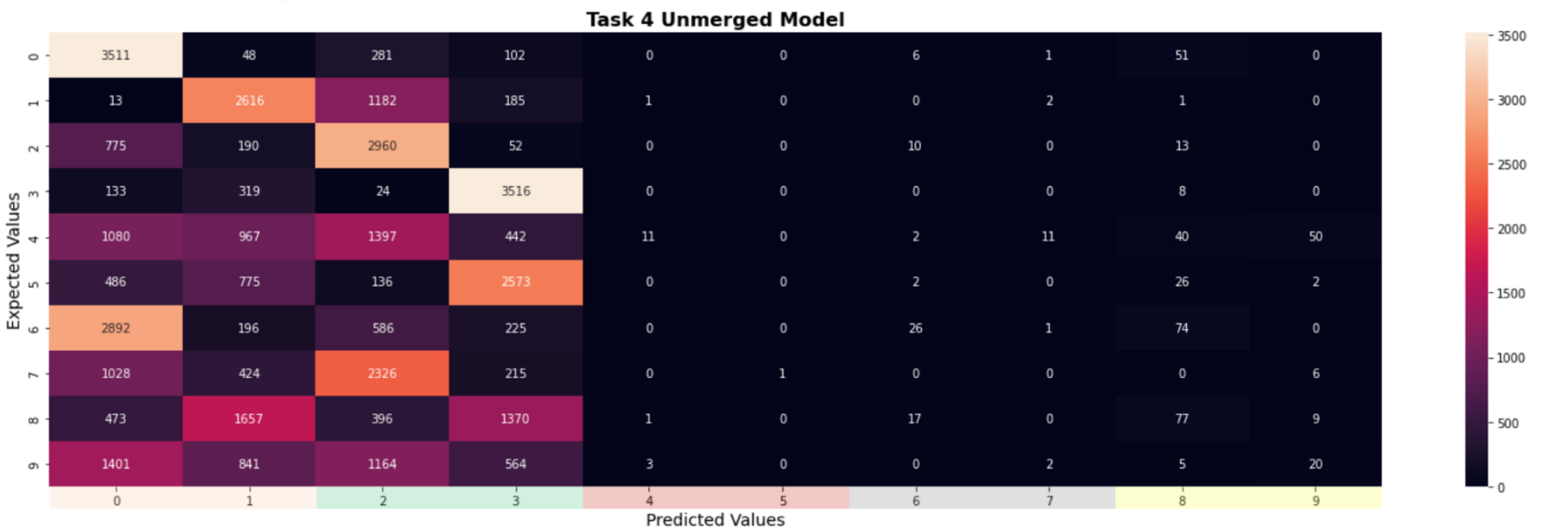


Figure 27: Confusion Matrix for Task 4 Unmerged Model on entire EMNIST dataset

TABLE VII: Performance Metrics of Task 4 Unmerged Model for EMNIST dataset

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	0	0.87775	0.297744	0.87775	0.444656	3511	8281	27719	489	unmerged_T4_backbone_all_heads
1	1	0.65400	0.325657	0.65400	0.434804	2616	5417	30583	1384	unmerged_T4_backbone_all_heads
2	2	0.74000	0.283199	0.74000	0.409632	2960	7492	28508	1040	unmerged_T4_backbone_all_heads
3	3	0.87900	0.380355	0.87900	0.530957	3516	5728	30272	484	unmerged_T4_backbone_all_heads
4	4	0.00275	0.687500	0.00275	0.005478	11	5	35995	3989	unmerged_T4_backbone_all_heads
5	5	0.00000	0.000000	0.00000	0.000000	0	1	35999	4000	unmerged_T4_backbone_all_heads
6	6	0.00650	0.412698	0.00650	0.012798	26	37	35963	3974	unmerged_T4_backbone_all_heads
7	7	0.00000	0.000000	0.00000	0.000000	0	17	35983	4000	unmerged_T4_backbone_all_heads
8	8	0.01925	0.261017	0.01925	0.035856	77	218	35782	3923	unmerged_T4_backbone_all_heads
9	9	0.00500	0.229885	0.00500	0.009787	20	67	35933	3980	unmerged_T4_backbone_all_heads

Based on the confusion matrix and performance metrics of task 4 unmerged model, only 26 out of 4000 samples were correctly classified as 6 and no

samples were classified as 7 even though the dataset contained 4000 images that contained the digit 7. Just as the observation made with task 3 unmerged model, 3511 samples out of 4000 have been classified as 0 and 3516 samples out of 4000 have been classified as 3. Task 4 model was trained on classifying 6 and 7, however, it performed poorly on classifying 6 and 7 correctly compared to its classification of 0 and 3. This indicates that the output scores for classes 0 and 3 may be higher than the output scores for 4 and 5 and model isn't well calibrated to identify 6 and 7.

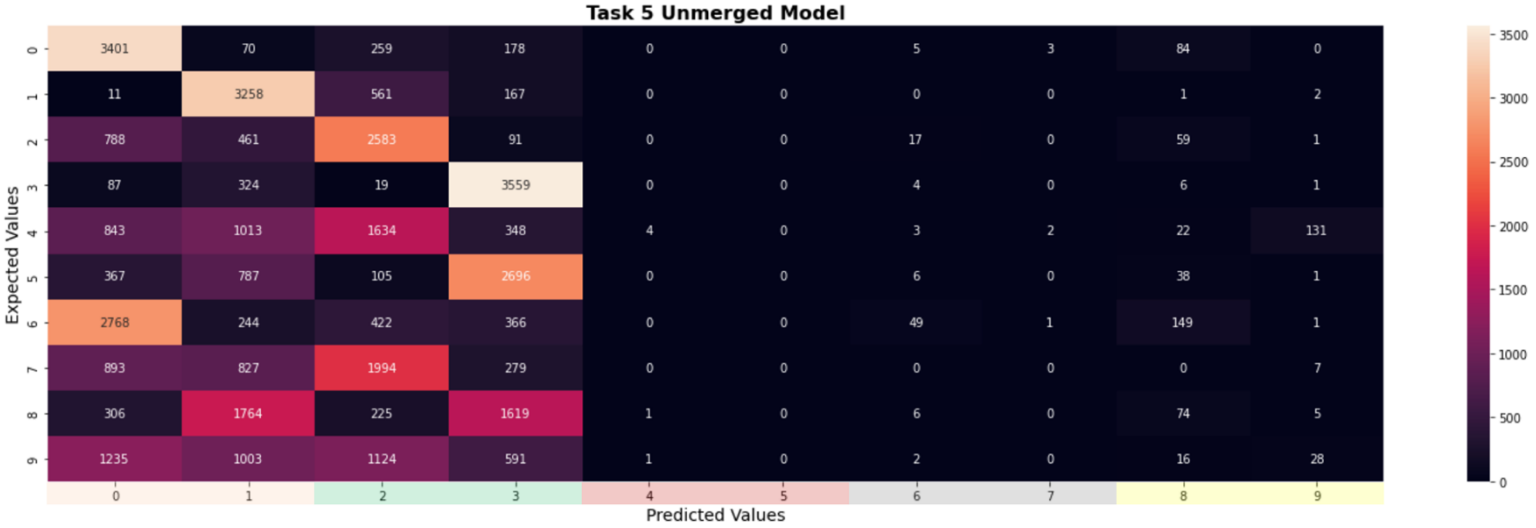


Figure 28: Confusion Matrix for Task 5 Unmerged Model on entire EMNIST dataset

TABLE VIII: Performance Metrics of Task 5 Unmerged Model for EMNIST dataset

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	0	0.85025	0.317880	0.85025	0.462753	3401	7298	28702	599	unmerged_T5_backbone_all_heads
1	1	0.81450	0.334120	0.81450	0.473856	3258	6493	29507	742	unmerged_T5_backbone_all_heads
2	2	0.64575	0.289379	0.64575	0.399660	2583	6343	29657	1417	unmerged_T5_backbone_all_heads
3	3	0.88975	0.359713	0.88975	0.512308	3559	6335	29665	441	unmerged_T5_backbone_all_heads
4	4	0.00100	0.666667	0.00100	0.001997	4	2	35998	3996	unmerged_T5_backbone_all_heads
5	5	0.00000	0.000000	0.00000	0.000000	0	0	36000	4000	unmerged_T5_backbone_all_heads
6	6	0.01225	0.532609	0.01225	0.023949	49	43	35957	3951	unmerged_T5_backbone_all_heads
7	7	0.00000	0.000000	0.00000	0.000000	0	6	35994	4000	unmerged_T5_backbone_all_heads
8	8	0.01850	0.164811	0.01850	0.033266	74	375	35625	3926	unmerged_T5_backbone_all_heads
9	9	0.00700	0.158192	0.00700	0.013407	28	149	35851	3972	unmerged_T5_backbone_all_heads

Based on the confusion matrix and performance metrics of task 5 unmerged model, only 74 out of 4000 samples were correctly classified as 8 and 28 samples out of 4000 were classified as 9. Just as the previous observations, there is high accuracy for the classification of 0 and 3. Task 5 model was trained on classifying 8 and 9, however, it performed poorly on classifying 8 and 9 correctly compared to its classification of 0 and 3. This indicates that the output scores for classes 0 and 3 may be higher than the output scores for 8 and 9 and the model isn't well calibrated to identify 8 and 9. Fig. 29 displays the plot of average accuracies received for each unmerged model on the entire EMNIST dataset. The low accuracy values for all unmerged models indicate that the models are not well calibrated to identifying the data they were trained on.

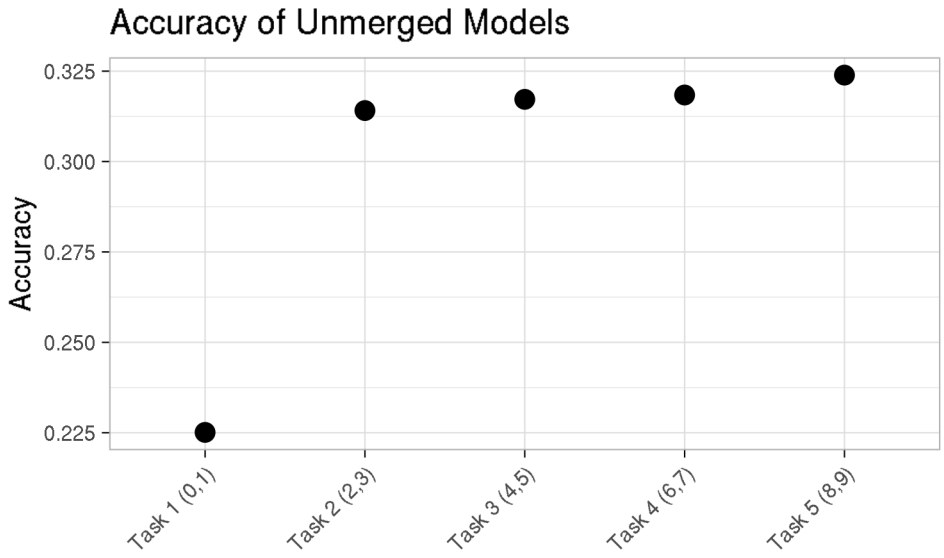


Figure 29: Plot of average accuracies for all 5 unmerged models with updated classifier layers for each model

iv. Merged Models with Classifier Layers of all Models

This is the same as the third robustness check that was done on unmerged models. Fig. 30 and Table IX contains the confusion matrix and performance metrics for task 2 merged model, respectively. Fig. 31 and Table X contains the confusion matrix and performance metrics for task 3 merged model, respectively. Fig. 32 and Table XI contains the confusion matrix and performance metrics for task 4 merged model, respectively. Fig. 33 and Table XII contains the confusion matrix and performance metrics for task 5 merged model, respectively.

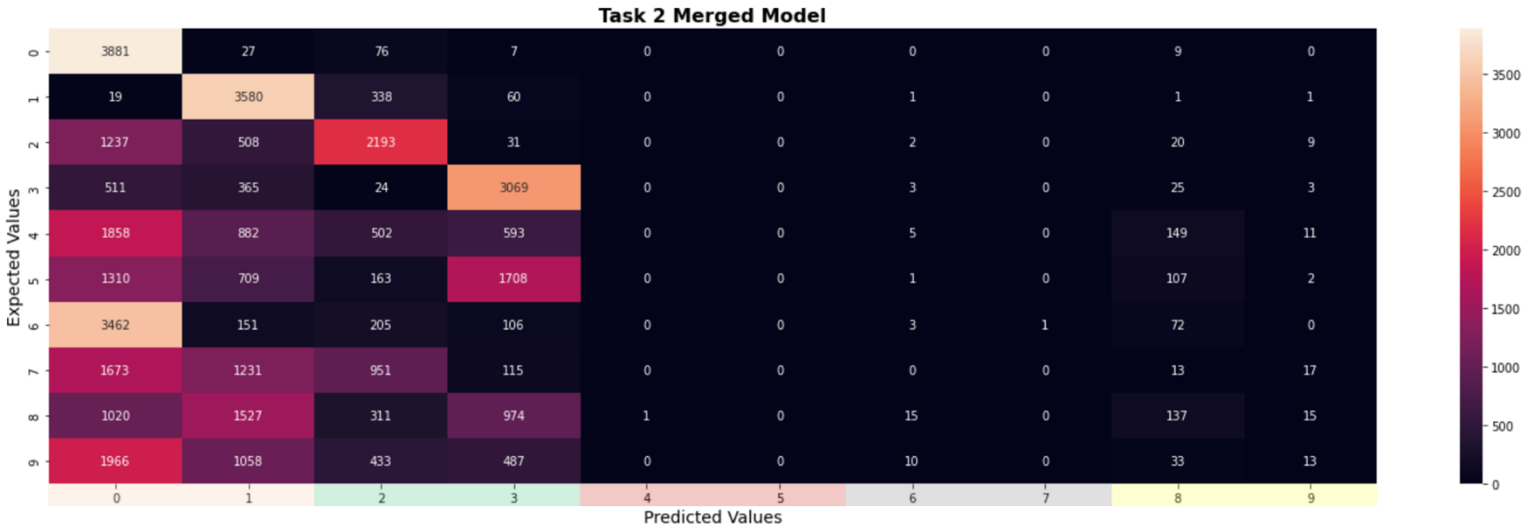


Figure 30: Confusion Matrix for Task 2 Merged Model on entire EMNIST dataset

TABLE IX: Performance Metrics of Task 2 Merged Model for EMNIST dataset

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	0	0.97025	0.229143	0.97025	0.370731	3881	13056	22944	119	merged_T2_backbone_all_heads
1	1	0.89500	0.356645	0.89500	0.510044	3580	6458	29542	420	merged_T2_backbone_all_heads
2	2	0.54825	0.422055	0.54825	0.476947	2193	3003	32997	1807	merged_T2_backbone_all_heads
3	3	0.76725	0.429231	0.76725	0.550493	3069	4081	31919	931	merged_T2_backbone_all_heads
4	4	0.00000	0.000000	0.00000	0.000000	0	1	35999	4000	merged_T2_backbone_all_heads
5	5	0.00000	0.000000	0.00000	0.000000	0	0	36000	4000	merged_T2_backbone_all_heads
6	6	0.00075	0.075000	0.00075	0.001485	3	37	35963	3997	merged_T2_backbone_all_heads
7	7	0.00000	0.000000	0.00000	0.000000	0	1	35999	4000	merged_T2_backbone_all_heads
8	8	0.03425	0.242049	0.03425	0.060009	137	429	35571	3863	merged_T2_backbone_all_heads
9	9	0.00325	0.183099	0.00325	0.006387	13	58	35942	3987	merged_T2_backbone_all_heads

The weights of the task 2 merged model are the average of the weights of task 1 and task 2 models. Based on the confusion matrix and performance metrics of

task 2 merged model, 3881 samples out of 4000 were classified as 0, 3580 samples out of 4000 were classified as 1, 2193 samples out of 4000 were classified as 2 and 3069 samples out of 4000 were classified as 3. Even though the accuracy for identifying 0 and 1 is high, there is a low accuracy for identifying 2 and 3. This indicates that the model hasn't learned the important features to identify 2 and 3 well enough to be able to classify them correctly.

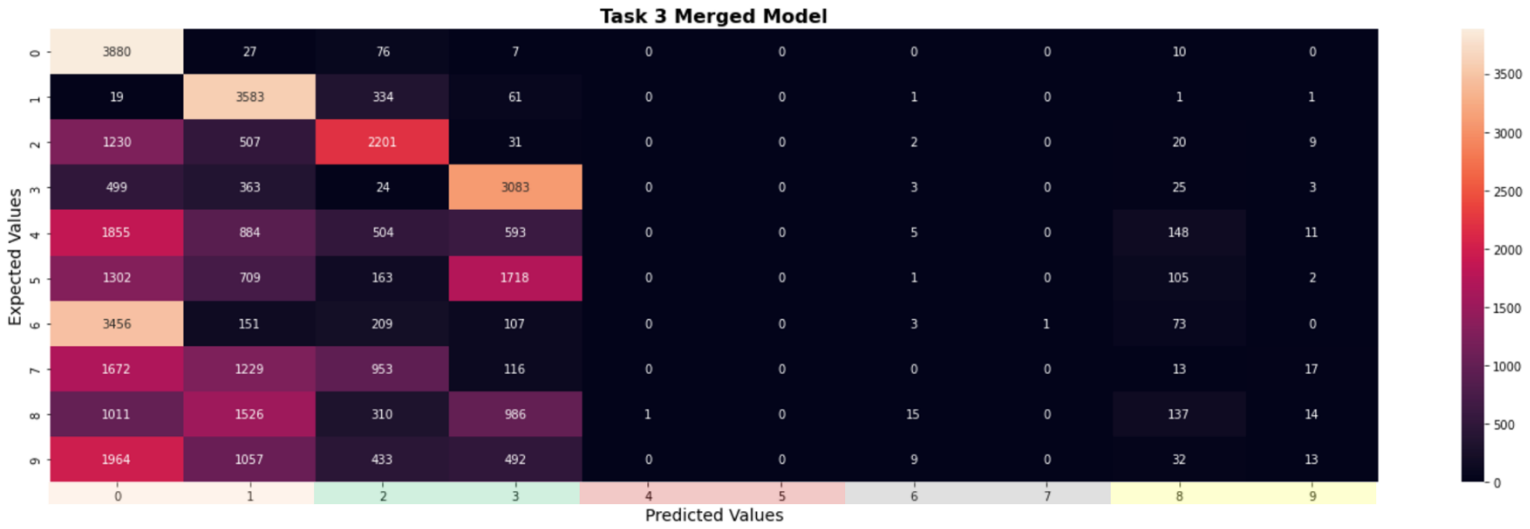


Figure 31: Confusion Matrix for Task 3 Merged Model on entire EMNIST dataset

TABLE X: Performance Metrics of Task 3 Merged Model

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	0	0.97000	0.229749	0.97000	0.371505	3880	13008	22992	120	merged_T3_backbone_all_heads
1	1	0.89575	0.357015	0.89575	0.510544	3583	6453	29547	417	merged_T3_backbone_all_heads
2	2	0.55025	0.422700	0.55025	0.478114	2201	3006	32994	1799	merged_T3_backbone_all_heads
3	3	0.77075	0.428552	0.77075	0.550831	3083	4111	31889	917	merged_T3_backbone_all_heads
4	4	0.00000	0.000000	0.00000	0.000000	0	1	35999	4000	merged_T3_backbone_all_heads
5	5	0.00000	0.000000	0.00000	0.000000	0	0	36000	4000	merged_T3_backbone_all_heads
6	6	0.00075	0.076923	0.00075	0.001486	3	36	35964	3997	merged_T3_backbone_all_heads
7	7	0.00000	0.000000	0.00000	0.000000	0	1	35999	4000	merged_T3_backbone_all_heads
8	8	0.03425	0.242908	0.03425	0.060035	137	427	35573	3863	merged_T3_backbone_all_heads
9	9	0.00325	0.185714	0.00325	0.006388	13	57	35943	3987	merged_T3_backbone_all_heads

The weights of the task 3 merged model are the average of the weights of task 1, task 2 and task 3 models. Based on the confusion matrix and performance metrics of task 3 merged model, 3880 samples out of 4000 were classified as 0, 3583 samples out of 4000 were classified as 1, 2201 samples out of 4000 were classified as 2 and 3083 samples out of 4000 were classified as 3. Task 3 model has been trained on digits 4 and 5, however, task 3 merged model is unable to classify digits 4 and 5. This indicates that the model hasn't learned the important features to identify 4 and 5 well enough to be able to classify them correctly.

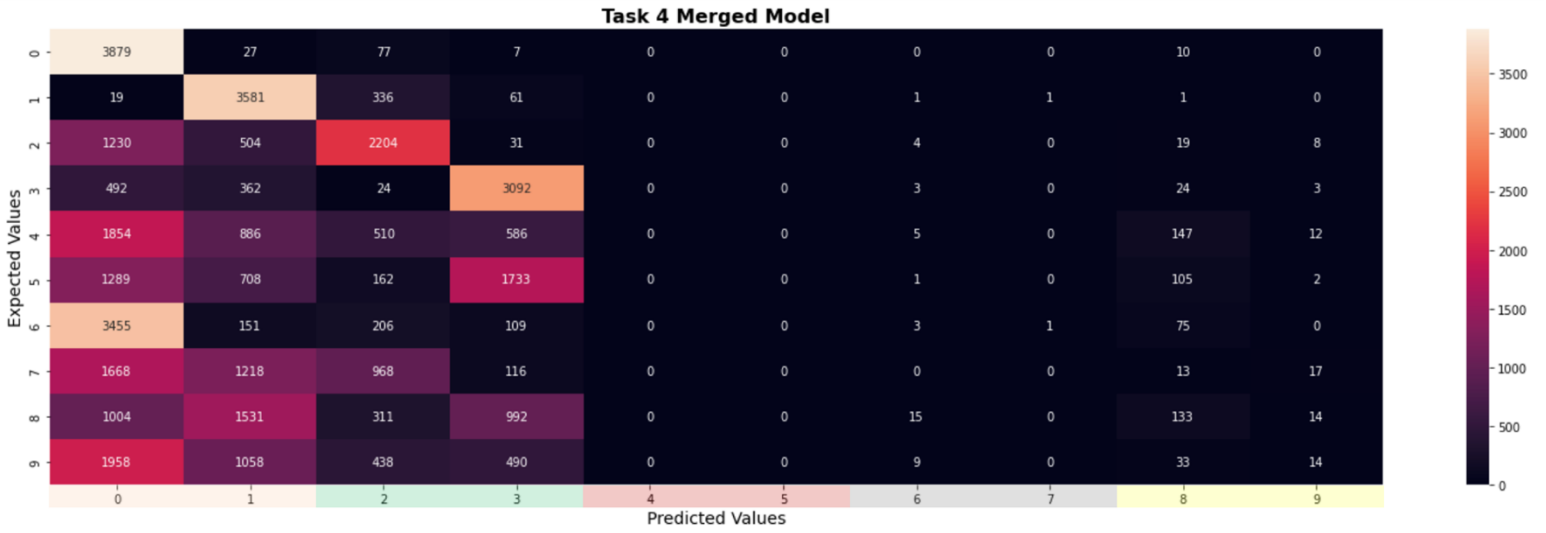


Figure 32: Confusion Matrix for Task 4 Merged Model on entire EMNIST dataset

TABLE XI: Performance Metrics of Task 4 Merged Model

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	0	0.96975	0.230235	0.96975	0.372122	3879	12969	23031	121	merged_T4_backbone_all_heads
1	1	0.89525	0.357171	0.89525	0.510623	3581	6445	29555	419	merged_T4_backbone_all_heads
2	2	0.55100	0.420932	0.55100	0.477263	2204	3032	32968	1796	merged_T4_backbone_all_heads
3	3	0.77300	0.428433	0.77300	0.551306	3092	4125	31875	908	merged_T4_backbone_all_heads
4	4	0.00000	0.000000	0.00000	0.000000	0	0	36000	4000	merged_T4_backbone_all_heads
5	5	0.00000	0.000000	0.00000	0.000000	0	0	36000	4000	merged_T4_backbone_all_heads
6	6	0.00075	0.073171	0.00075	0.001485	3	38	35962	3997	merged_T4_backbone_all_heads
7	7	0.00000	0.000000	0.00000	0.000000	0	2	35998	4000	merged_T4_backbone_all_heads
8	8	0.03325	0.237500	0.03325	0.058333	133	427	35573	3867	merged_T4_backbone_all_heads
9	9	0.00350	0.200000	0.00350	0.006880	14	56	35944	3986	merged_T4_backbone_all_heads

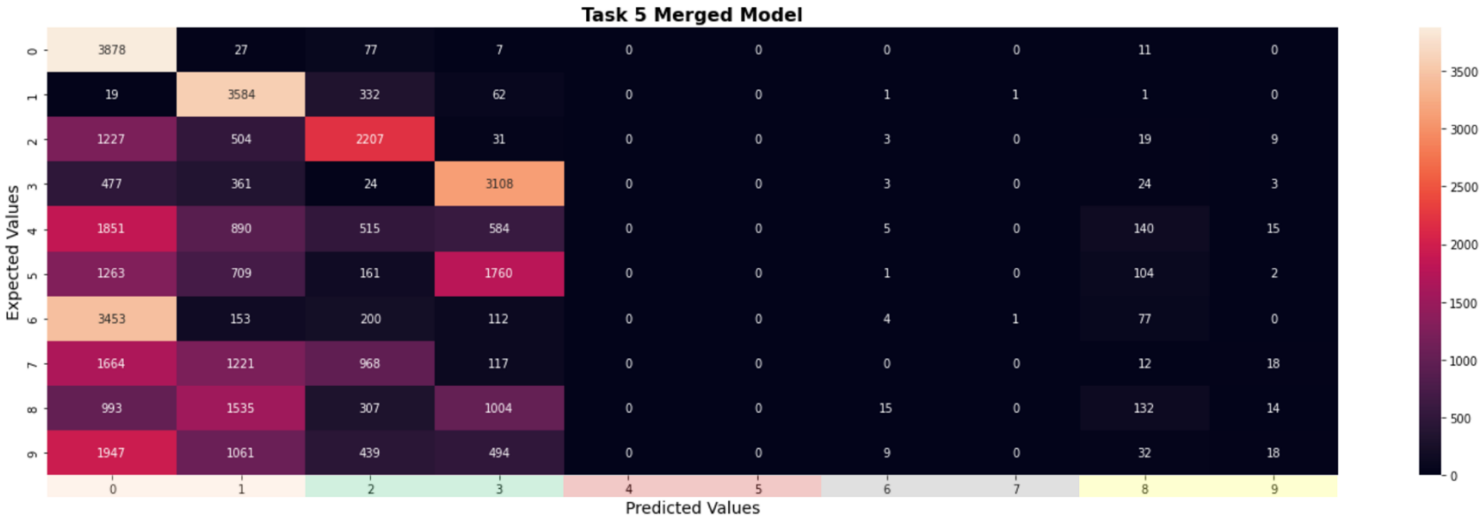


Figure 33: Confusion Matrix for Task 5 Merged Model on entire EMNIST dataset

TABLE XII: Performance Metrics of Task 5 Merged Model

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	0	0.96950	0.231219	0.96950	0.373387	3878	12894	23106	122	merged_T5_backbone_all_heads
1	1	0.89600	0.356794	0.89600	0.510360	3584	6461	29539	416	merged_T5_backbone_all_heads
2	2	0.55175	0.421989	0.55175	0.478223	2207	3023	32977	1793	merged_T5_backbone_all_heads
3	3	0.77700	0.426982	0.77700	0.551113	3108	4171	31829	892	merged_T5_backbone_all_heads
4	4	0.00000	0.000000	0.00000	0.000000	0	0	36000	4000	merged_T5_backbone_all_heads
5	5	0.00000	0.000000	0.00000	0.000000	0	0	36000	4000	merged_T5_backbone_all_heads
6	6	0.00100	0.097561	0.00100	0.001980	4	37	35963	3996	merged_T5_backbone_all_heads
7	7	0.00000	0.000000	0.00000	0.000000	0	2	35998	4000	merged_T5_backbone_all_heads
8	8	0.03300	0.239130	0.03300	0.057996	132	420	35580	3868	merged_T5_backbone_all_heads
9	9	0.00450	0.227848	0.00450	0.008826	18	61	35939	3982	merged_T5_backbone_all_heads

The results of task 4 and task 5 merged models don't showcase any further improvement when compared to task 3 merged model results. Fig. 34 displays the plot of average accuracies received for each merged model on the

entire EMNIST dataset. These low accuracy values are expected since the unmerged model counterparts also show low accuracy values.

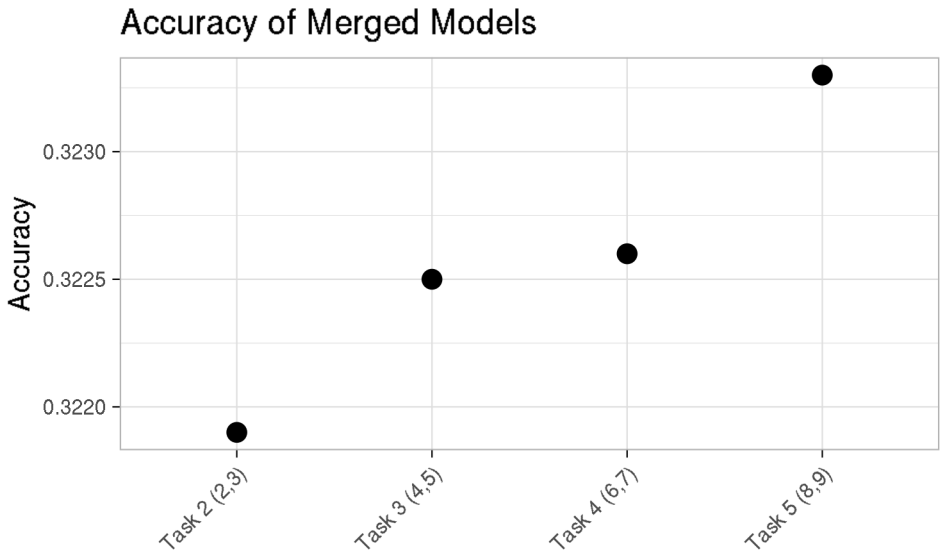


Figure 34: Plot of average accuracies of all 4 merged models

b) Robustness Check of MIT Indoor Scenes Experiment

The first robustness check that was done was to see how well all unmerged models performed on unseen task data. Appropriate task data was fed into corresponding task models in batches of 32 and an average accuracy was computed after all the data had been processed. The supplementary section of this report showcases the performance metrics of all unmerged models on unseen task data. Fig. 35 provides the average accuracy values for unmerged models that were received on the newly created scenes dataset. Based on the results, the average accuracy is low for all tasks on unseen data.

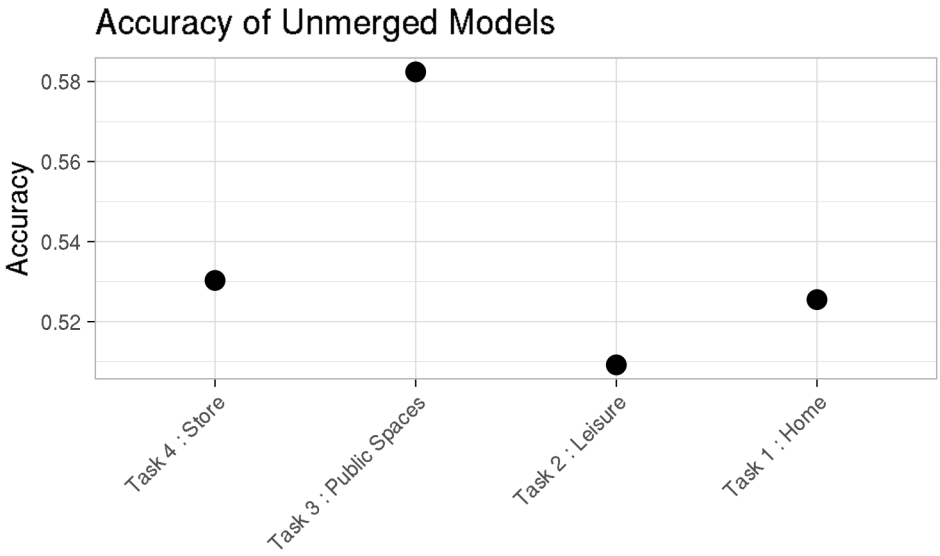


Figure 35: Plot of average accuracies of all 4 unmerged models

The second robustness check was to verify that each merged model performed correctly on its corresponding task. As a brief recap, each parameter in a merged model contains the mean parameter value of its previous tasks. Like how the first robustness check was done, appropriate task data was fed into corresponding task merged models in batches of 32 and an average accuracy was computed after all the data had been processed. The supplementary section of this report showcases the performance metrics of all merged models on unseen task data Fig. 36 provides the average accuracy values for all merged models that were received on the new scenes' dataset for iteration 5.

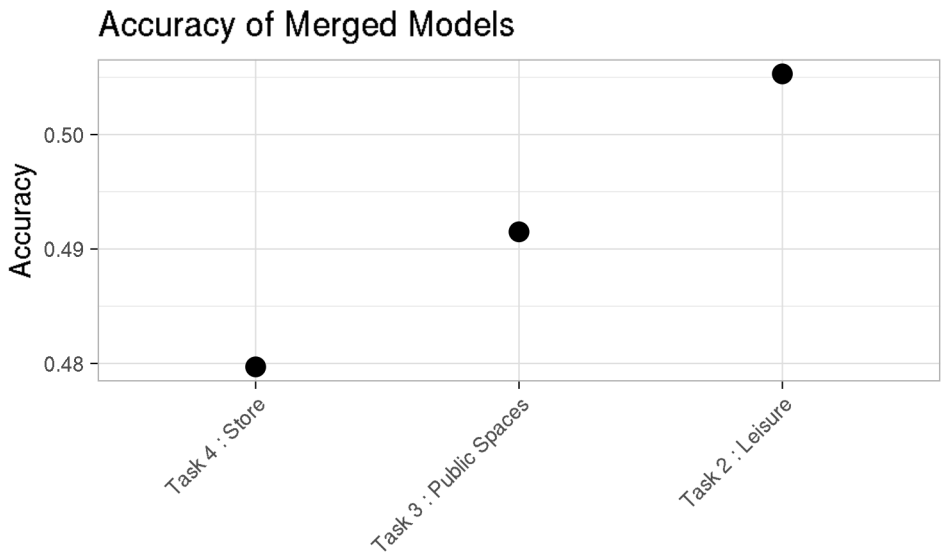


Figure 36: Plot of average accuracies of all 3 merged models

According to Fig. 36, the average accuracy for task 2 merged model is approximately the same compared to its unmerged counterpart in Fig. 30. The average accuracies for task 3 and task 4 merged models reduced significantly compared to its unmerged counterparts. This indicates that information may have been lost during the merging process to correctly identify the appropriate class.

Additional experiments were performed to check for robustness. The DUA framework was not compared with any other federated learning frameworks or baseline for the MIT indoor scenes experiment. Based on this observation, a centralized model, FedAvg [7], and FedProx [15] were utilized to create a baseline for the DUA framework for the MIT indoor scenes experiment.

The MIT indoor scenes dataset was used in three different data distributions, homogeneous, heterogeneous, and random. In homogeneous data distribution, each user is assigned the same number of images depending upon the total number of images and number of users. For example, if there are 10 images and 5 users, each user gets assigned 2 images. In heterogeneous data distribution, each user will get the same fraction of images from all categories. For example, user 1 is allocated 30% of images from store, home, public and leisure super categories from the MIT indoor scene dataset. In random data distribution, a user can get a random number of images from a random category and this data distribution closely resembles a real-world setting. In all three data distributions, each user has unique images. All three data distributions were utilized both in FedAvg [7] and FedProx [15] frameworks.

To implement federated learning environments for indoor scenes predictions, the FedAvg and FedProx implementations from the FedMA GitHub repo [18] was used with minor modifications. The MIT indoor scenes dataset was partitioned between 5 users using all three data distribution methods. Compared to the DUA framework, the implementations of FedAvg [7] and FedProx [15] included the working place category in the MIT indoor scenes dataset.

i. *Homogeneous Data Distribution*: Each user received 20% of images from each category in the MIT indoor scenes dataset.

ii. *Heterogeneous Data Distribution:* Even though each user is allocated images from all categories, the proportion of images that each user receives from all categories is different. For example, user 1 receives 20% of images from all categories, user 2 receives 30% of images from all categories, user 3 receives 10% of images from all categories, user 4 receives 15% of images from all categories, and lastly, user 5 receives 25% of images from all categories.

iii. *Random Data Distribution:* Each user receives a random number of images from any category in the MIT indoor scenes dataset.

The same VGG11 architecture that was used in the DUA framework [13] for the indoor scenes experiments was used to build the model. Each user model was initialized with this VGG11 model with a few frozen layers. Layers till the penultimate convolutional layer ('features 16') were frozen to preserve the low-level features learned from ImageNet. Each user model was trained on local data. The minibatch training involved 25 epochs while taking advantage of data augmentation due to the comparatively limited training dataset size. PyTorch implementations of cross entropy loss function and SGD were used for training.

Once the local training phase was completed, the FedAvg [7] and FedProx [15] were implemented using the user models. A notable change from the original implementation of these frameworks is that at each communication

round, instead of selecting a fraction of users at random for local training, all 5 users in this implementation participated in every communication round.

Fig. 37 showcases a plot comparing the average accuracies of all federated learning frameworks on the MIT indoor scene dataset. Based on the results shown in Fig. 37, FedProx [15] and FedAvg [7] with homogeneous data distribution showed the highest accuracy among all the federated learning frameworks, including DUA. Data distribution and user personalization differentiate DUA from the rest of the frameworks which is why it's a comparable candidate with the rest of the frameworks.

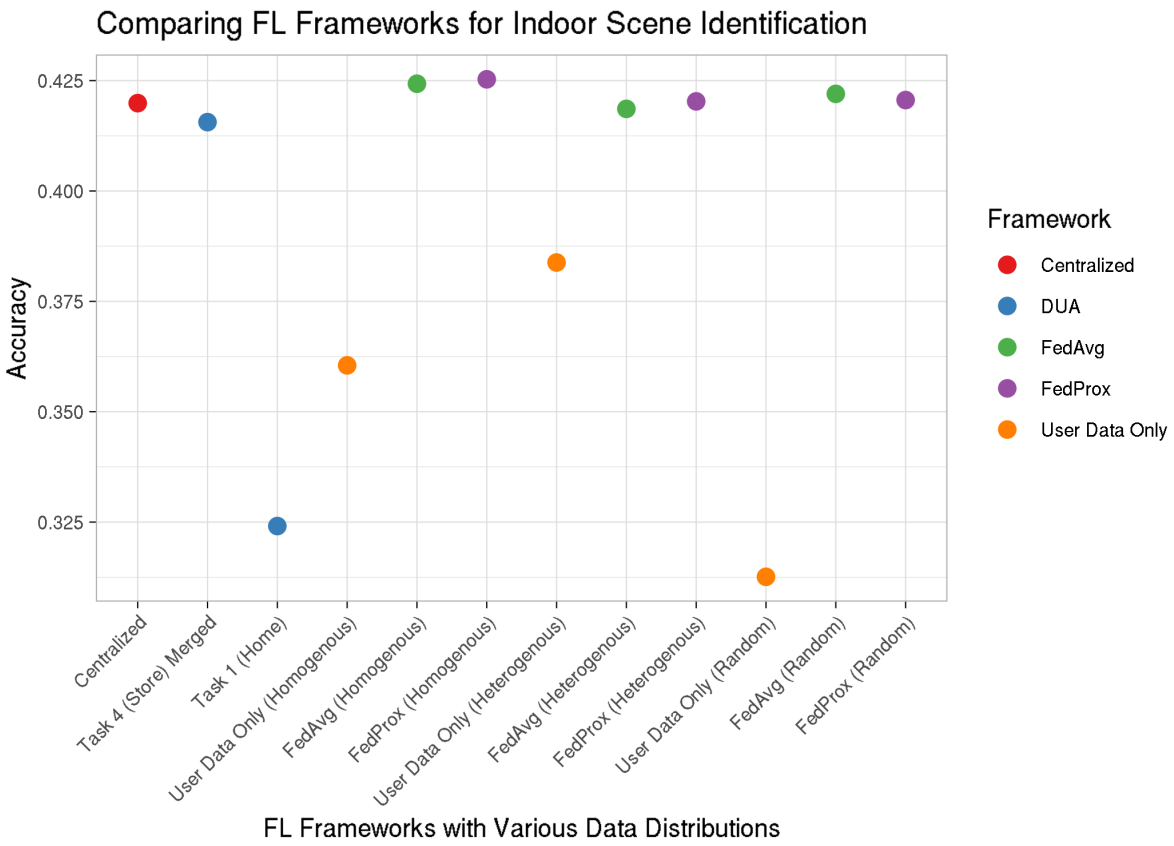


Figure 37: Plot of average accuracies for Federated Learning (FL) frameworks for Indoor Scene Recognition

VII. METHODOLOGY

Based on the results of the preliminary research that have been shown in the previous section, the DUA framework doesn't appear to work well or consider new unseen tasks. In a real-world setting, user data is constantly evolving. In other words, there is no way to predict the data that a user device will contain, and it will be a violation of the user's privacy to know beforehand what the user data might be. For this purpose, it's important that the DUA framework considers unseen tasks so it can be adaptable to new data. With that being said, the goal of the proposed methodology is to address the issue of performance on unseen tasks by the DUA and improve its performance. There will be two phases in this research. Phase 1 will be focused on creating new tasks with overlapped datasets. In the original DUA experiment, the server trained models were trained on unique subsets of the entire dataset. No two models shared the same subset of training data. In our experiment, we decided to let two models share the same subset of data because the culmination of the features learned by each model can help to better identify tasks. After a set of new unmerged and merged models have been created, these models will be evaluated on unseen EMNIST data. In theory, the last merged model is trained for all previous tasks and current task. With that being said, the classifier layer of the last merged model will be replaced with the classifier layer of all the models on the assumption that the classifier layer of each model is trained well to identify the task it was trained on. If the last merged model is trained for all tasks and each classifier layer can identify the task it was trained on, any

data that is fed into this model configuration should predict tasks correctly. Phase 2 will be focused on reconstructing the last merged model into multiple configurations and retraining the new configured models to determine which model improved performance. As the preliminary research has shown, the findings on the number datasets experiments generalizes to the MIT indoor scenes dataset. Hence, the rest of this thesis will focus on the numbers experiment.

A. Implementation Plan for Numbers Experiment

1) Phase 1

1. Create tasks with overlapped datasets using MNIST and SVHN datasets as shown in Table XIII for both training on the server and collecting importance weights from user devices.

TABLE XIII: Tasks of Overlapped Datasets

TASKS	SUBSET OF DIGITS
1	0, 1
2	1, 2
3	2, 3
4	3, 4
5	4, 5
6	5, 6
7	6, 7
8	7, 8
9	8, 9
10	9, 0

2. Train each of the 10 tasks using the DUA framework resulting in 10 unmerged models.

3. Calculate the importance weights from the user data using the DUA framework
4. Merge each task model, except for the first one, resulting in a total of 9 merged models.
5. Pre-process the EMNIST digits dataset by applying a series of transforms.
6. Divide the EMNIST digits dataset into the same 10 tasks as defined in Table XIII.
7. Feed the unseen EMNIST digits task data into the server trained models and evaluate how each unmerged model performs on unseen data.
8. Repeat the previous step for the merged models for user 1 and evaluate how each merged model performs on unseen data.
9. Replace the head layer of each merged model with the head layer of all models.
10. Feed the unseen EMNIST digits data into each corresponding newly configured model and evaluate how each merged model performs on unseen data.

2) Phase 2

1. Create a new configuration (Configuration 1) for the merged model of task 10 by replacing its classifier layer with classifier layers of all the models as shown in Fig. 38.

a. Note that the classifier layer of the newly configured task 10 merged model will contain 20 output nodes.

2. Each digit in the range 0 to 9 contains 2 output nodes in the classifier layer. The average value of the 2 output nodes will be taken for each digit. There should be 10 output probabilities at the end. The maximum among these 10 output probabilities will be the prediction of the model. See Fig. 38.

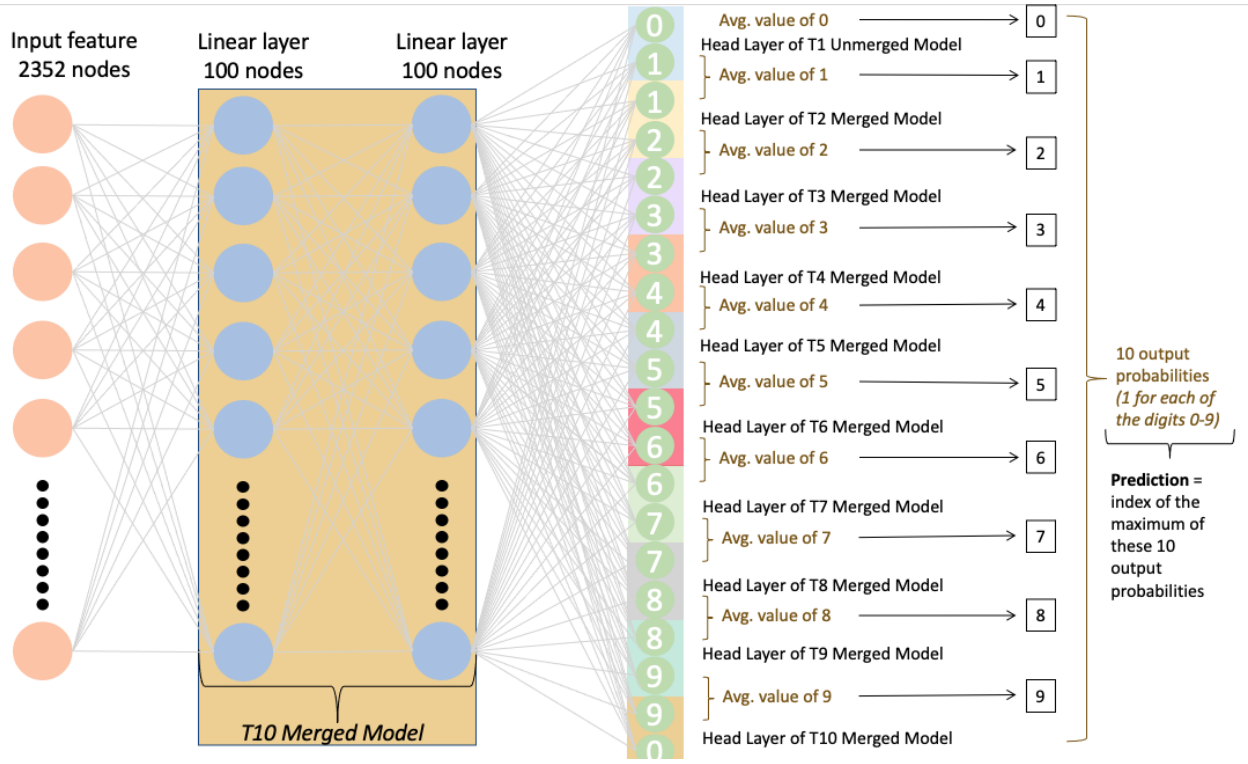


Figure 38: Model Configuration 1

3. Train only the head layer of Configuration 1 by freezing the parameters of task 10 merged model

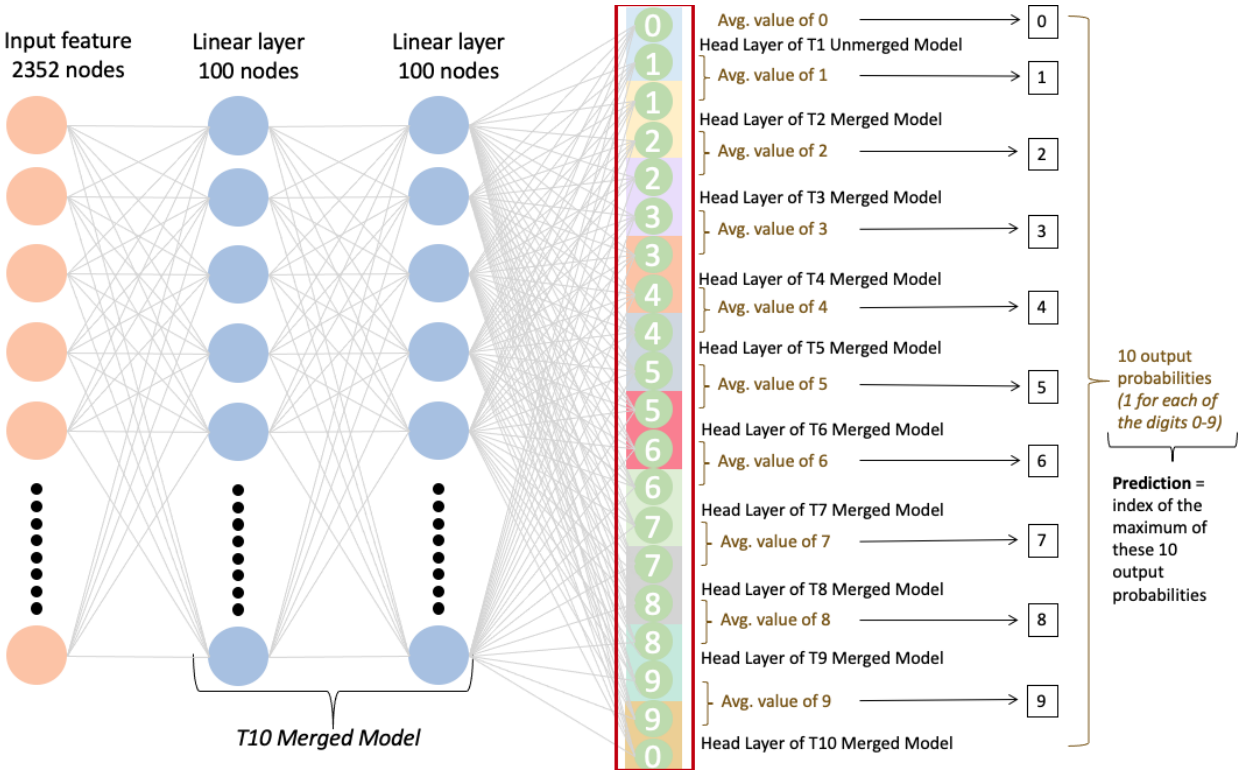


Figure 39: Training head layer only of model configuration 1

4. Configuration 2 is the same as configuration with only one difference. The maximum value of 2 output nodes will be taken for each digit. The maximum among the 10 output probabilities will be the prediction of the model. See Fig. 40.

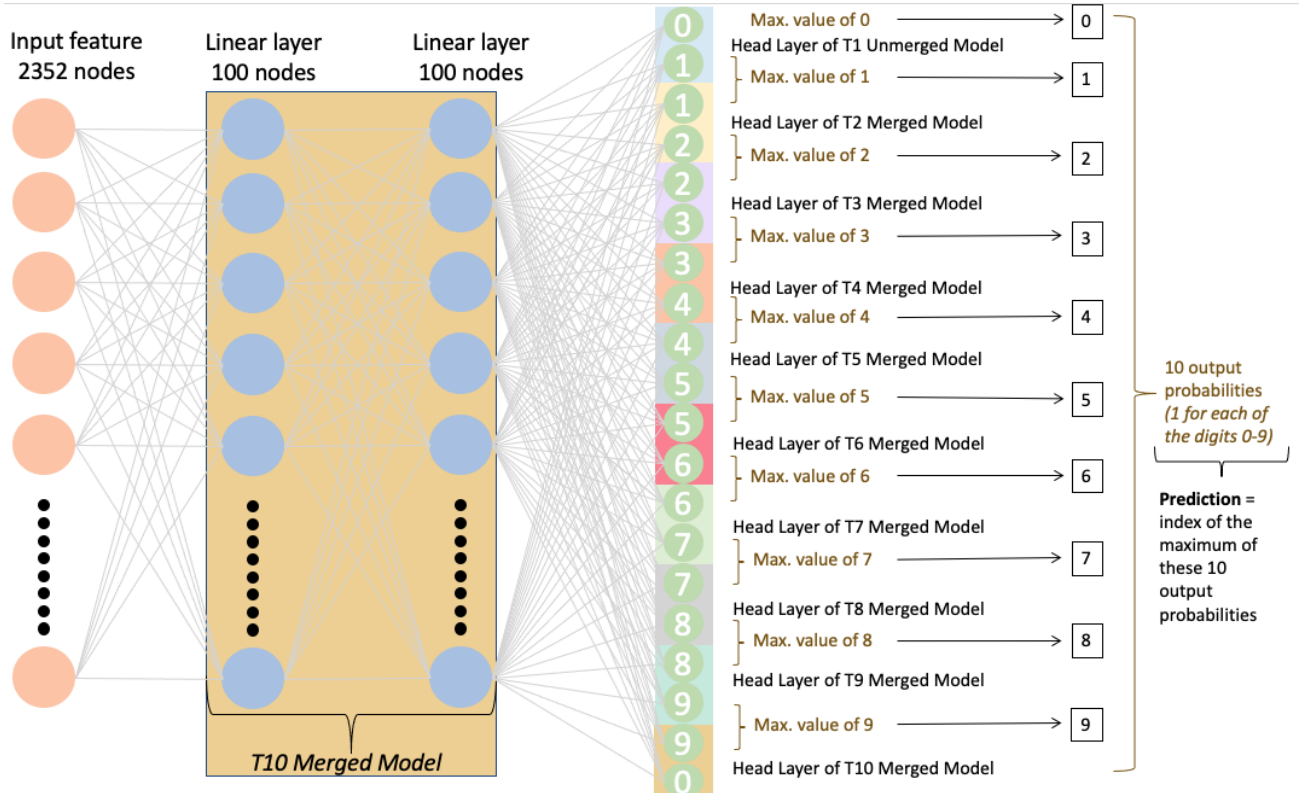


Figure 40: Model Configuration 2

5. Train only the head layer of Configuration 2 by freezing the parameters of task 10 merged model

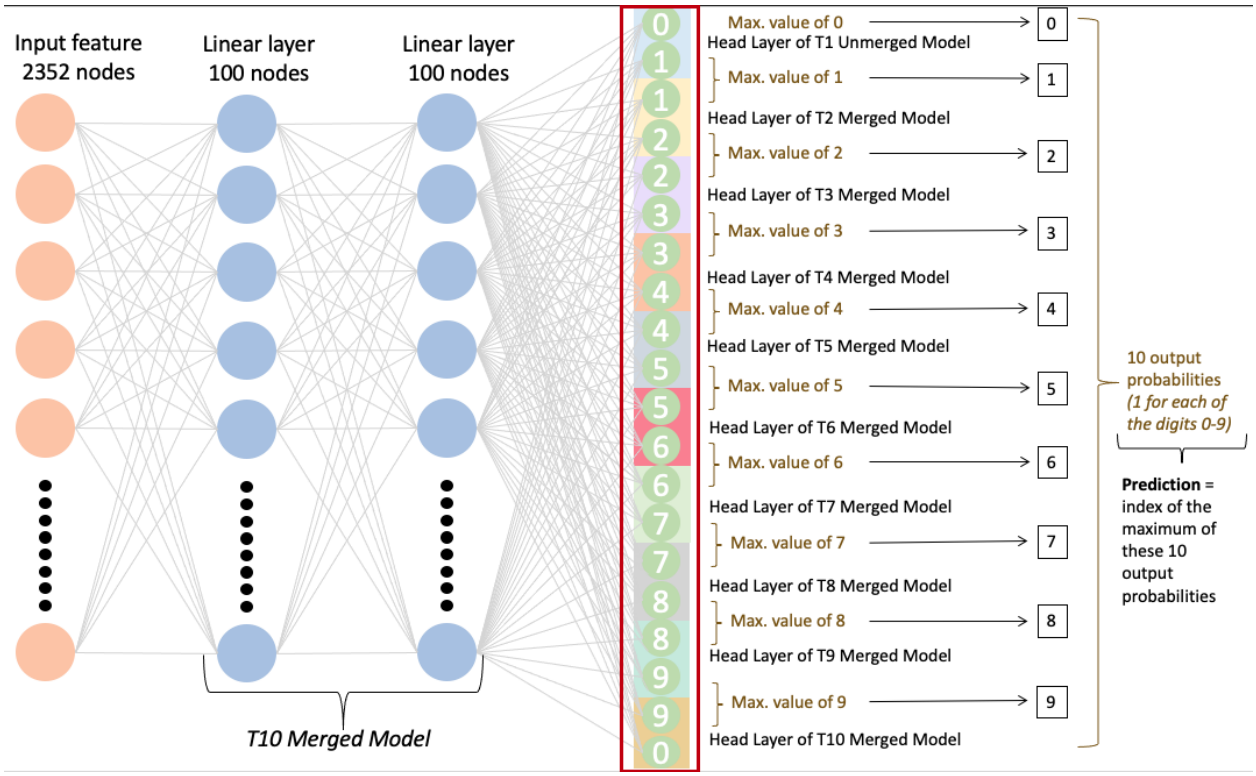


Figure 41: Training head layer only of model configuration 2

6. Configuration 3 for the merged model of task 10 is like Configuration 2 with one difference. Instead of taking the maximum value among two output nodes, Configuration 3 takes the sum of each class which results in 10 sum values. The maximum value among these 10 values is taken as the prediction. Configuration 3 is shown in Fig. 42.

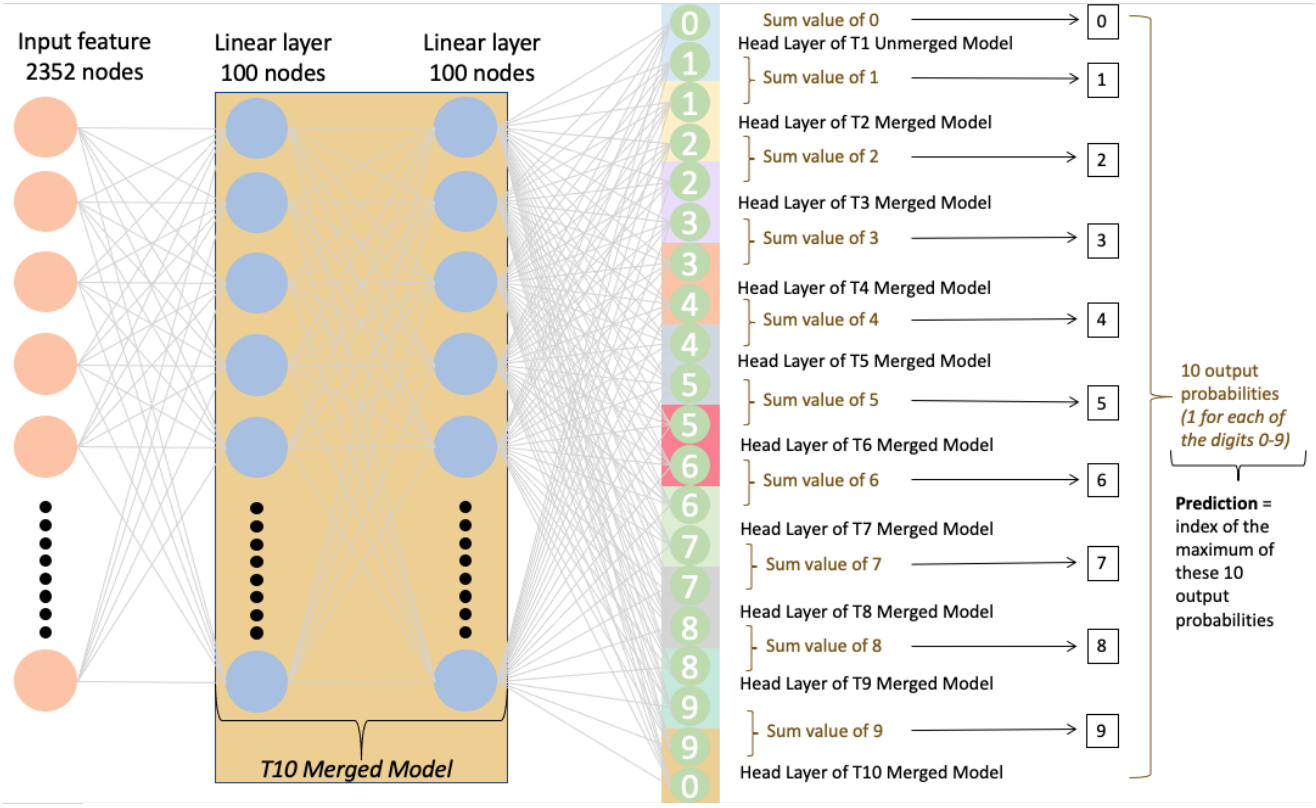


Figure 42: Model Configuration 3

7. Train only the head layer of Configuration 3 by freezing the parameters of task 10 merged model

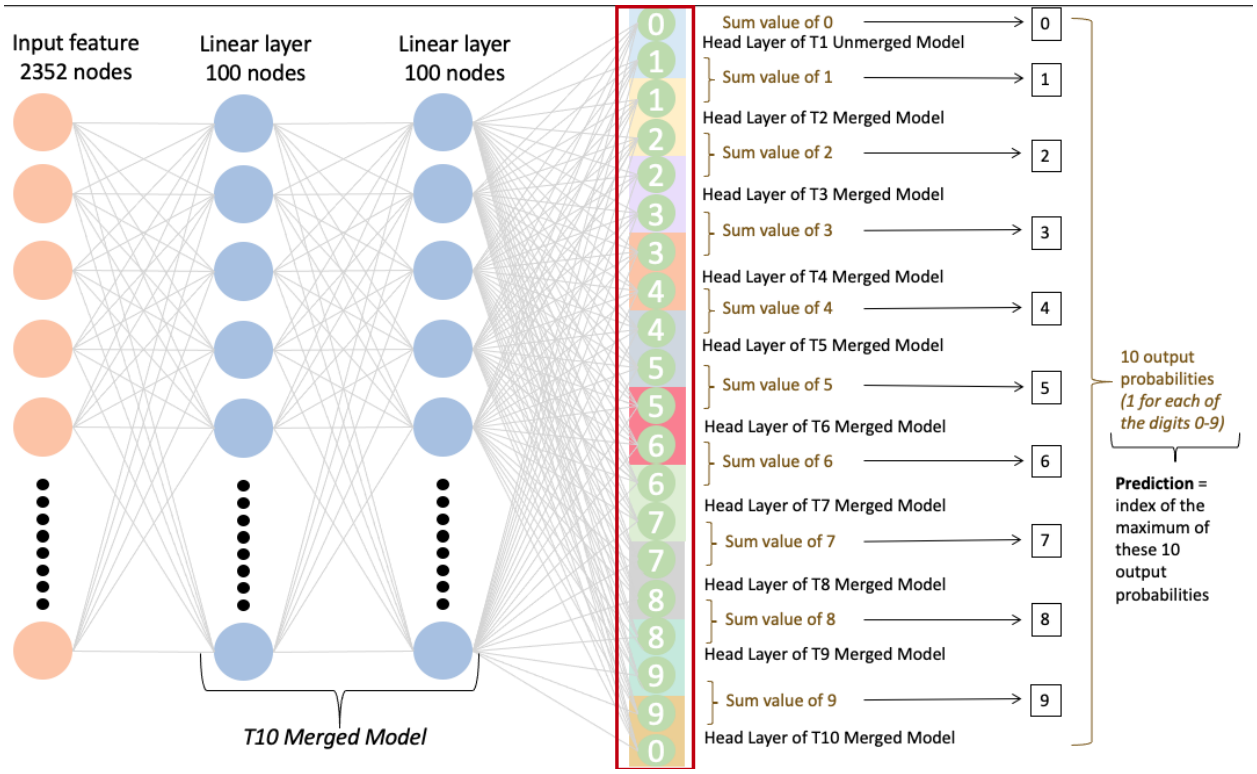


Figure 43: Training head layer only of model configuration 3

8. Create a new configuration (Configuration 4) for the merged model of task 10 by replacing its classifier layer with a new linear layer of 10 nodes as shown in Fig. 44.

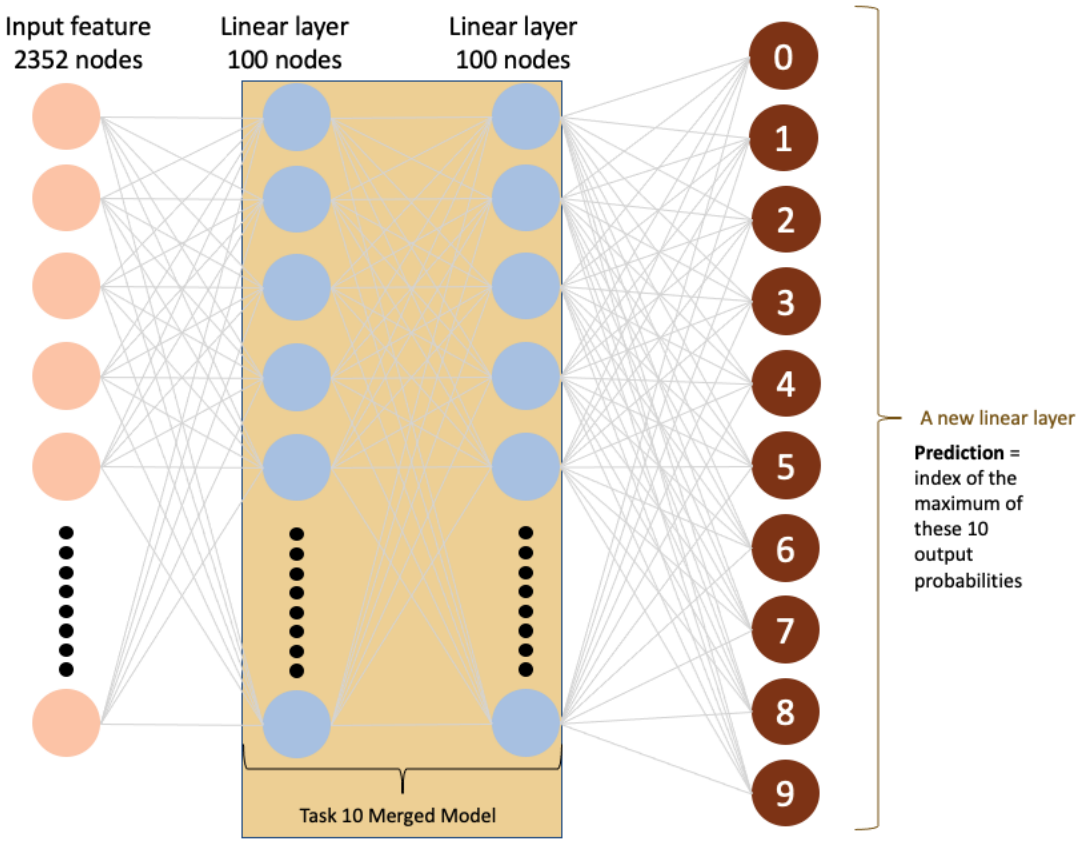


Figure 44: Model configuration 4

9. Train the final layer of Configuration 4 by freezing the parameters of the task 10 merged model

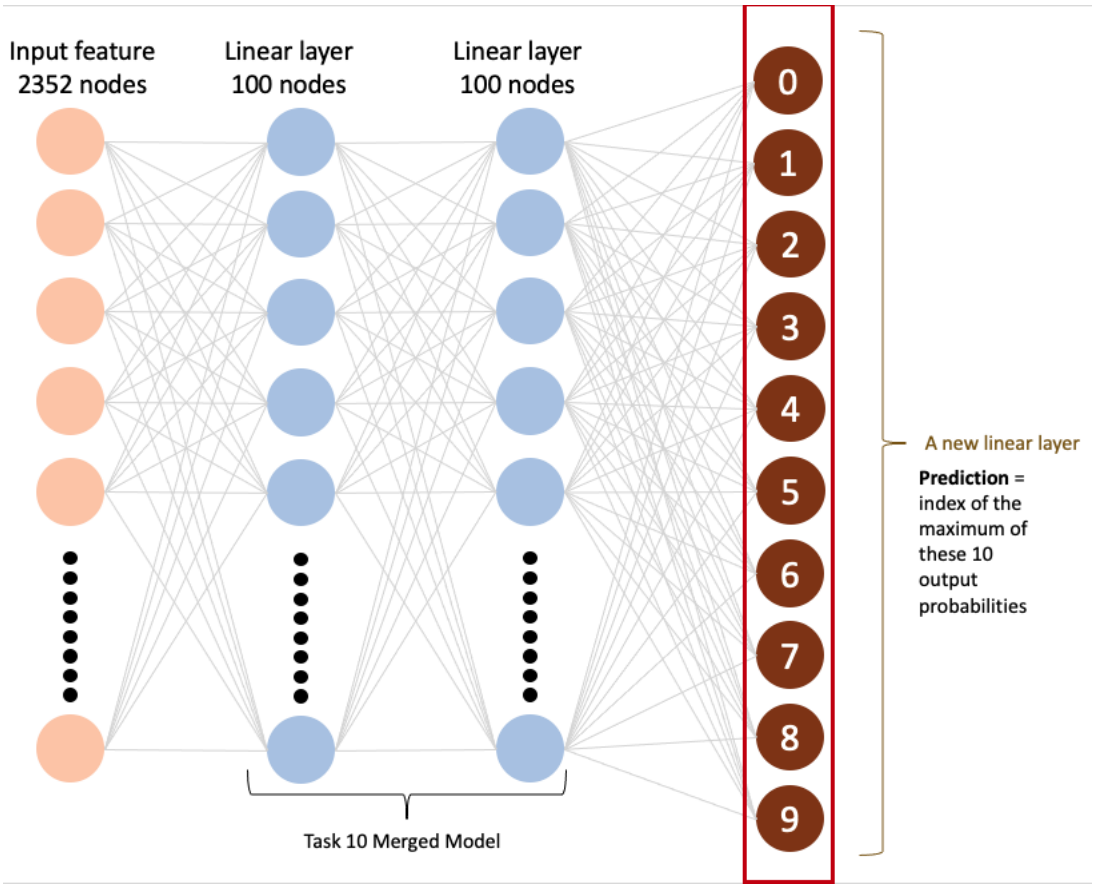


Figure 45: Training linear layer only of model configuration 4

10. Create a new configuration (Configuration 5) for the merged model of task 10 by replacing its classifier layer with the classifier layer of all models and inserting a new linear layer of 10 nodes as shown in Fig. 46.

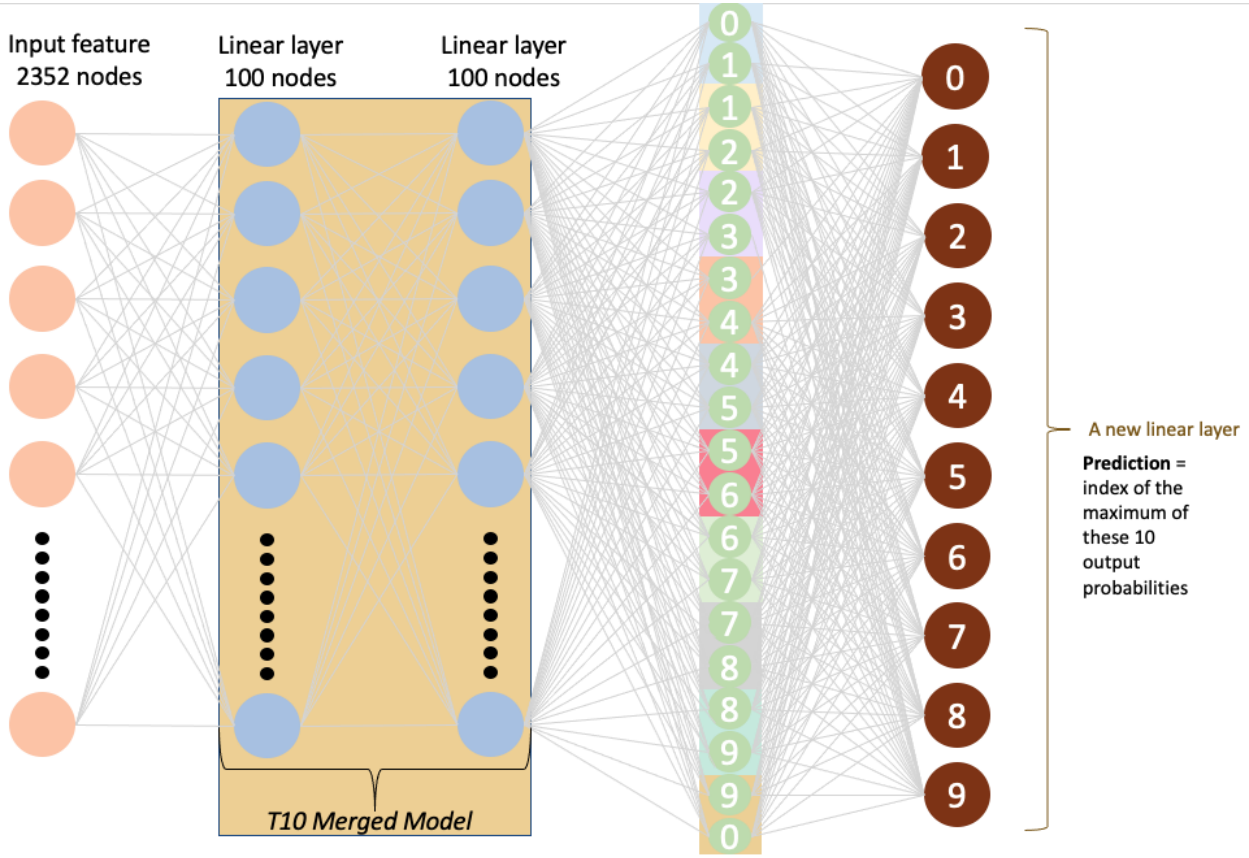


Figure 46: Model configuration 5

11. Train the final layer of Configuration 5 by freezing the parameters of the task 10 merged model and the head layer

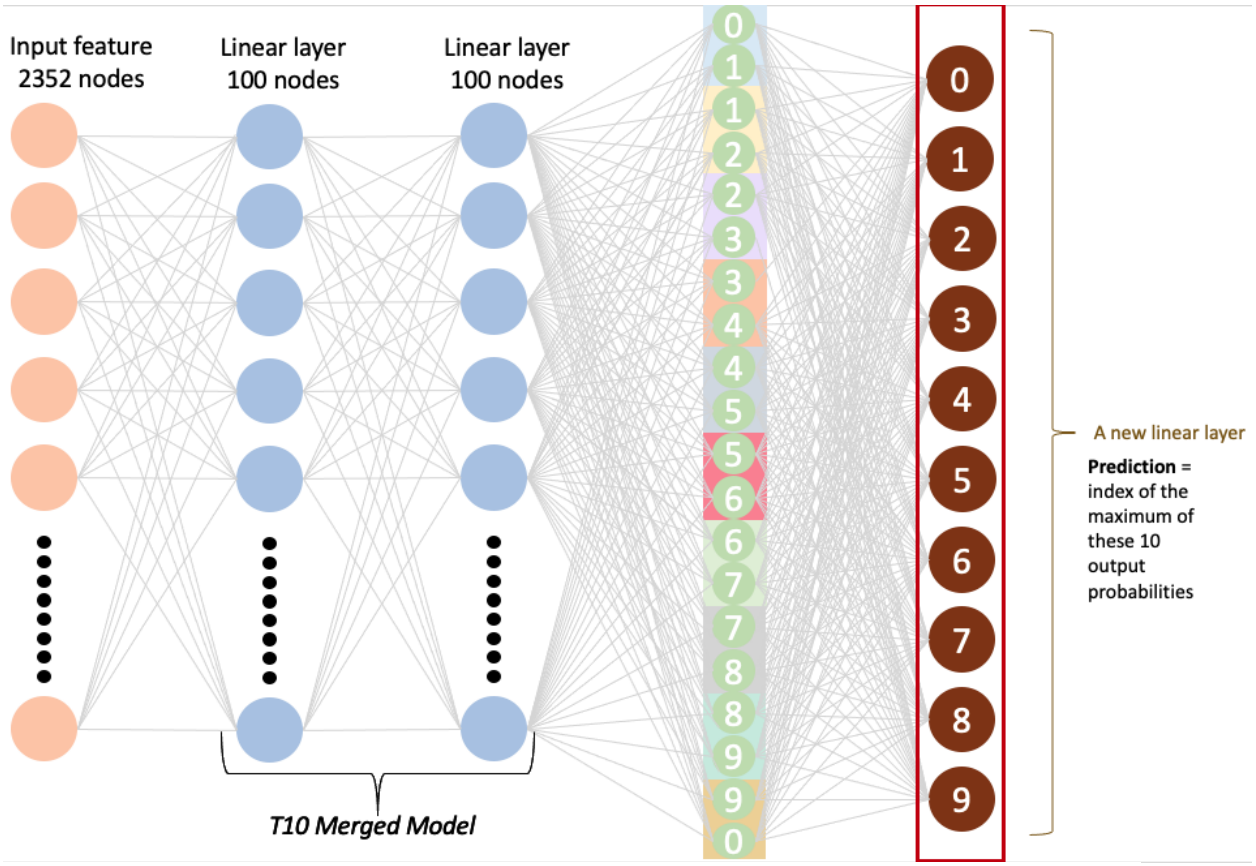


Figure 47: Training final layer only of model configuration 5

12. Train the final layer and head layer of Configuration 5 by freezing the parameters of the task 10 merged model (Configuration 6)

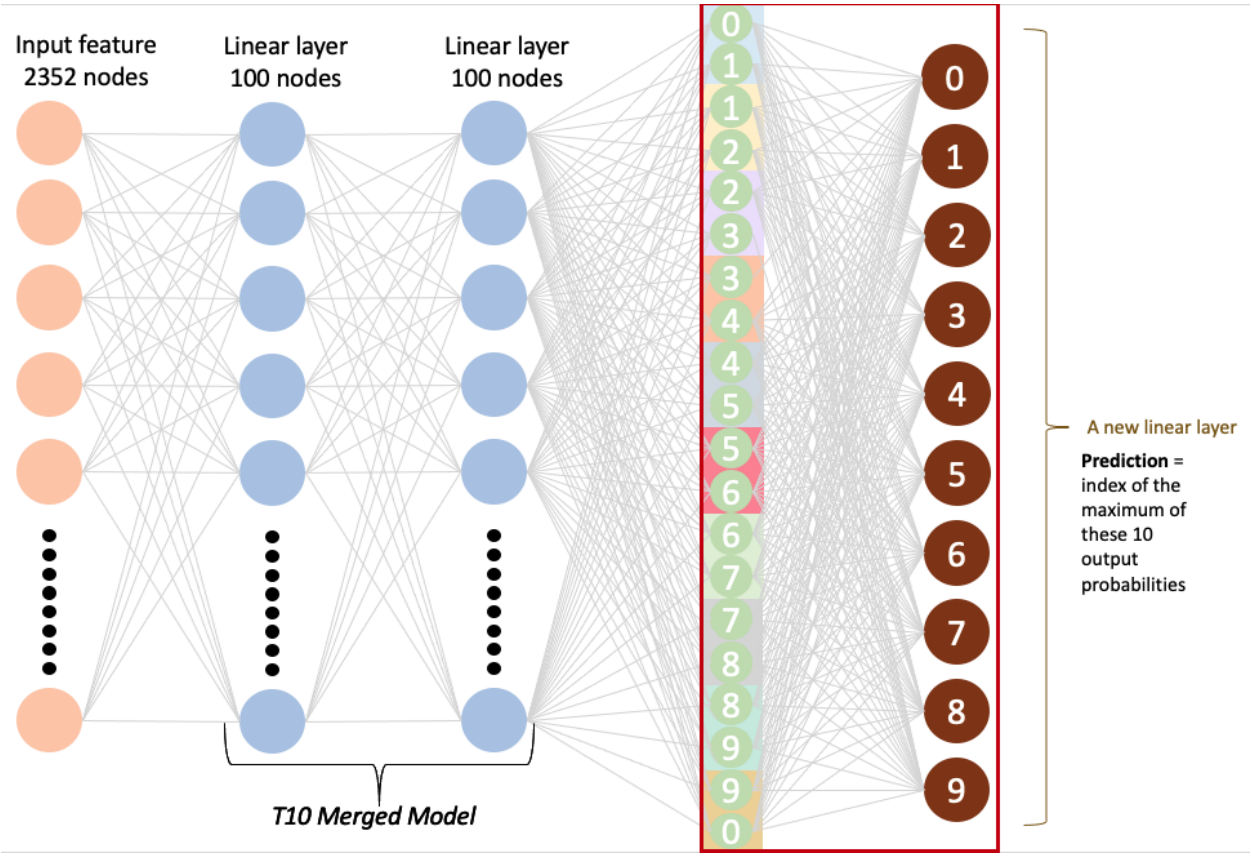


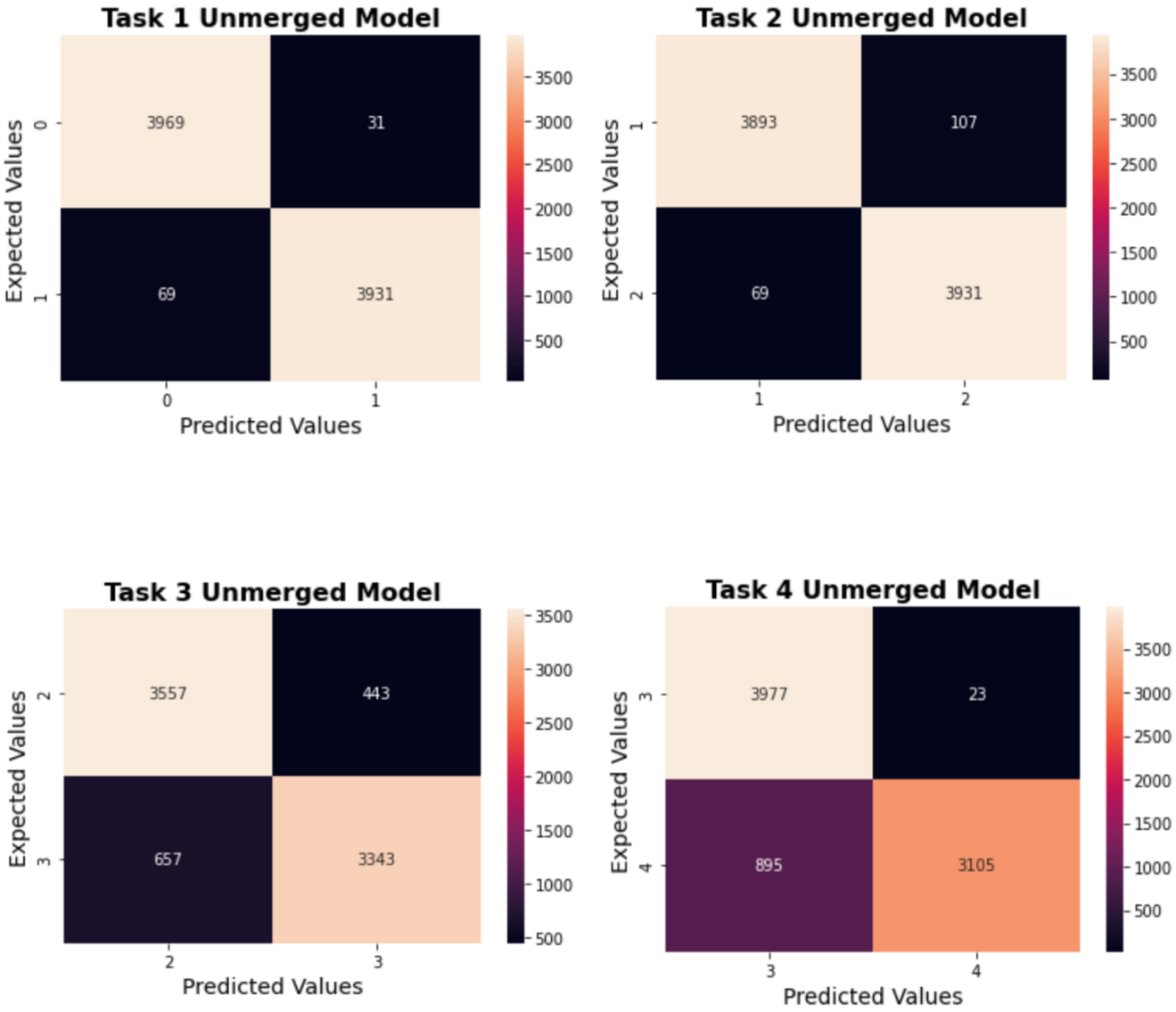
Figure 48: Training head and final layers only of model configuration 6

VIII. EXPERIMENTAL EVALUATION

A. *Evaluation Results of Numbers Experiment*

1) Phase 1: The training procedure of these overlapped tasks is the same as the training done on non-overlapped tasks. At the end of the execution of the DUA framework on overlapped datasets, there are 10 unmerged and 9 merged models. It's important to evaluate how these models work on unseen data prior to the rest of the experiment. The robustness of all these models will be evaluated in a similar way that was done on models trained on non-overlapped datasets. Fig. 49

showcases the confusion matrices for all 10 unmerged models followed by Fig, 50 which contains the average accuracies of all models.



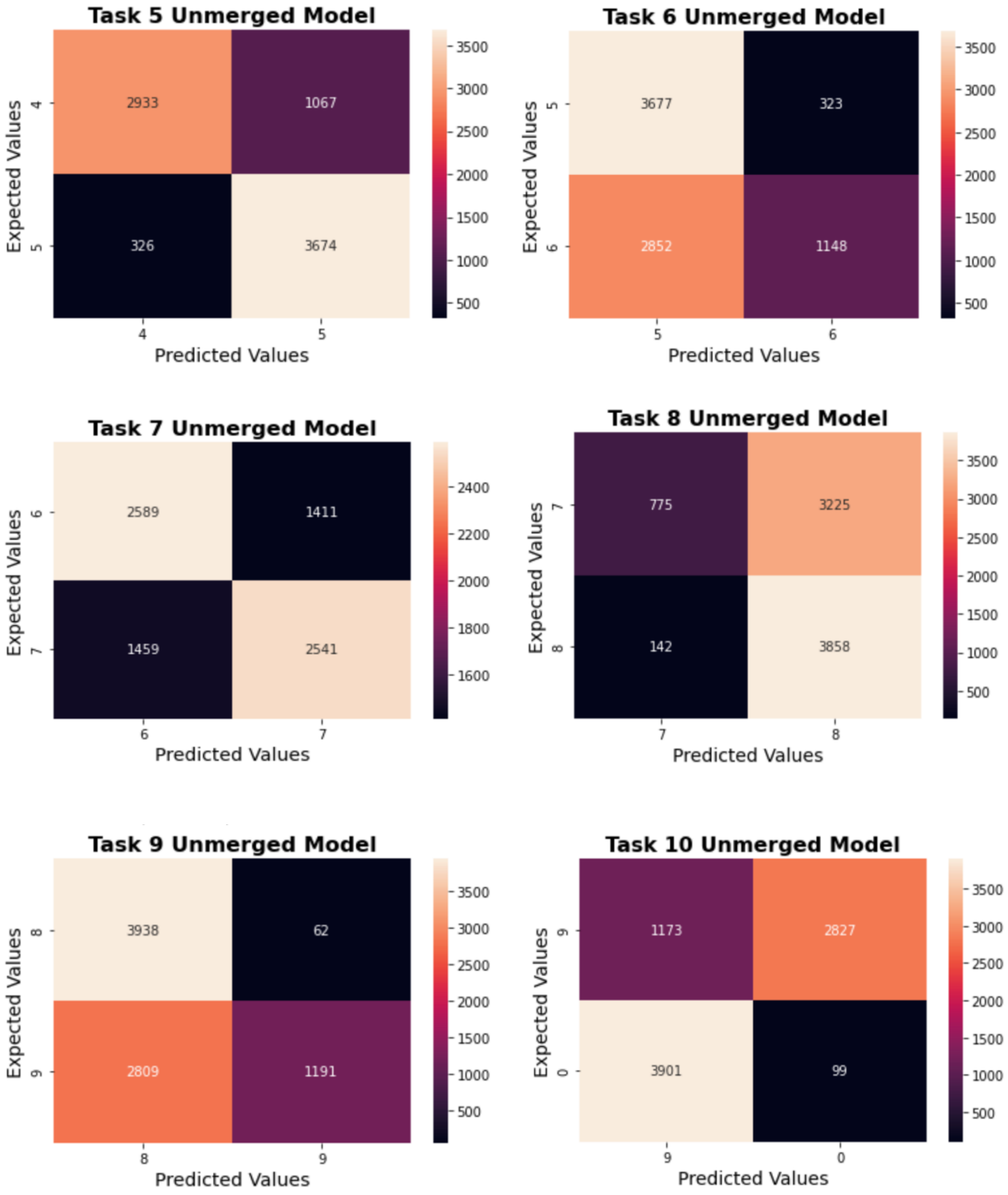


Figure 49: Confusion matrices of all 10 unmerged models

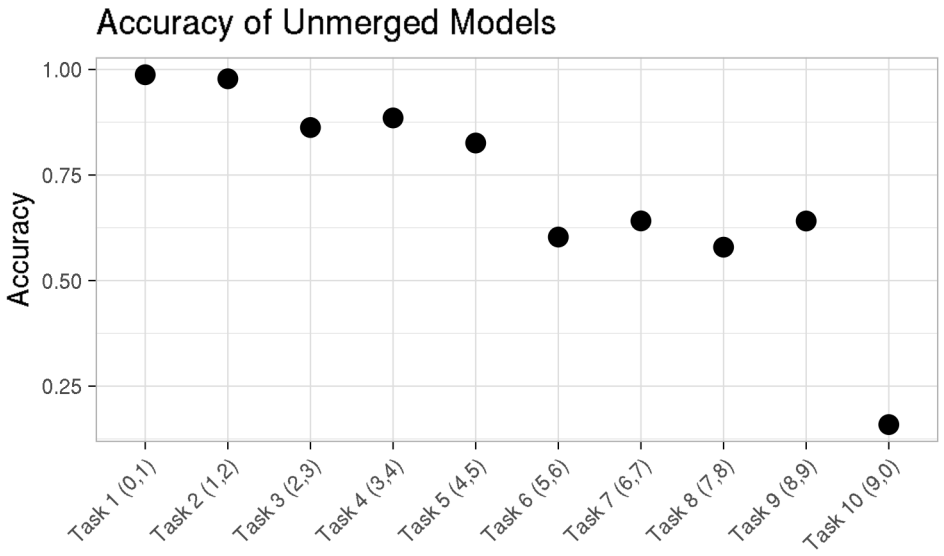


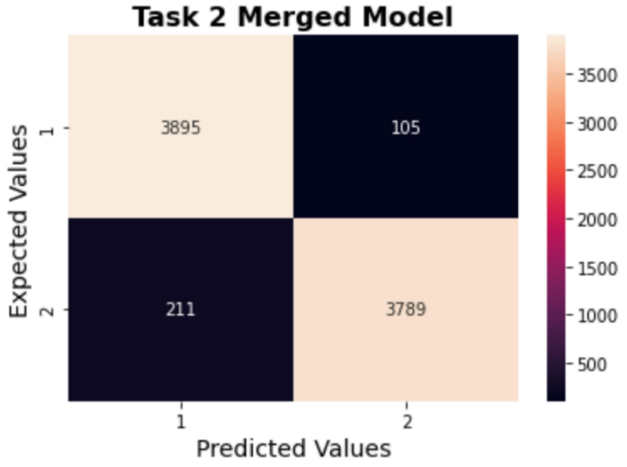
Figure 50: Plot of average accuracies of 10 unmerged models

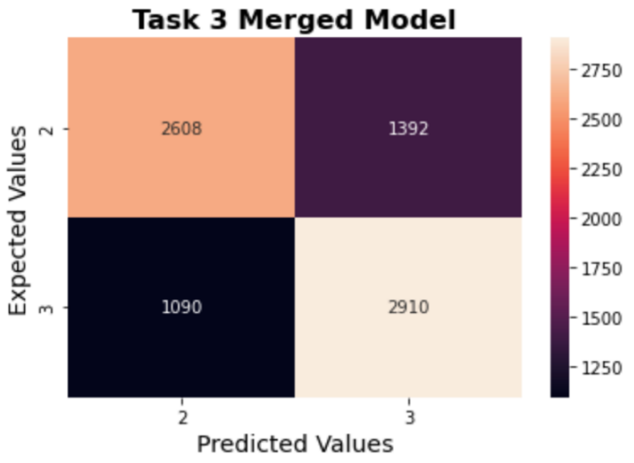
Based on the results shown in Fig. 49 and Fig. 50, unmerged models of task 1, task 2, task 3, task 4, and task 5 have a high accuracy of predicting the subset of digits it was trained on by testing it on corresponding unseen EMNIST dataset. A few key observations:

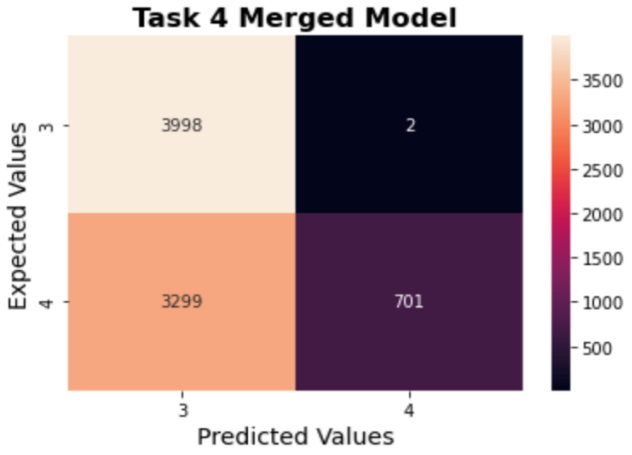
- 3969 samples were correctly classified as 0 by task 1 unmerged model whereas only 99 samples were correct classified as 0 by task 10 unmerged model
    - Indicates that task 10 unmerged model hasn't learned the general structure of identifying a digit as 0 or 9
- 1148 samples were correctly classified as 6 by task 6 unmerged model whereas 2589 samples were classified as 6 by task 7 unmerged model

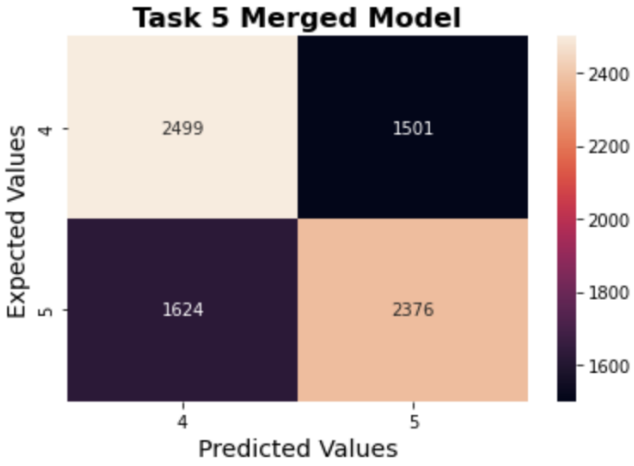
    - Indicates that task 6 unmerged model hasn't learned the general structure of identifying a digit as 5 or 6
    - Also, indicates that task 7 unmerged model has learned a better general structure of identifying a digit as 6 when model was trained on 6 and 7
- 2541 samples were correctly classified as 7 by task 7 unmerged model whereas only 775 samples were classified as 7 by task 8 unmerged model
    - Indicates that task 8 unmerged model hasn't learned the general structure of identifying a digit as 7 or 8

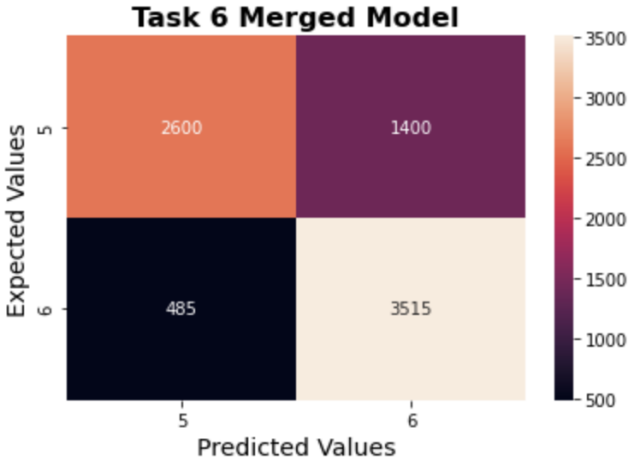
The same robustness check that was performed for unmerged models was also performed for merged models. Fig. 51 showcases the confusion matrices for all 9 merged models followed by Fig. 52 which contains the average accuracies of all models.

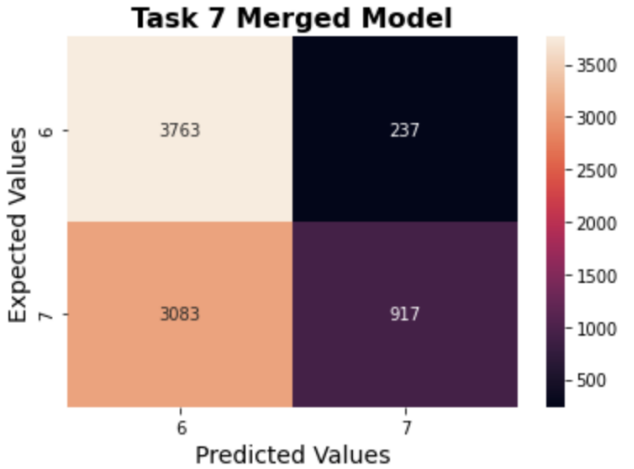

Task 2 Merged Model
3895
105
211
3789
Expected Values
Predicted Values
1
2
3500
3000
2500
2000
1500
1000
500


Task 3 Merged Model
2608
1392
1090
2910
Expected Values
Predicted Values
2
3
2750
2500
2250
2000
1750
1500
1250


Task 4 Merged Model
3998
2
3299
701
Expected Values
Predicted Values
3
4
3500
3000
2500
2000
1500
1000
500


Task 5 Merged Model
2499
1501
1624
2376
Expected Values
Predicted Values
4
5
2400
2200
2000
1800
1600


Task 6 Merged Model
2600
1400
485
3515
Expected Values
Predicted Values
5
6
3500
3000
2500
2000
1500
1000
500


Task 7 Merged Model
3763
237
3083
917
Expected Values
Predicted Values
6
7
3500
3000
2500
2000
1500
1000
500

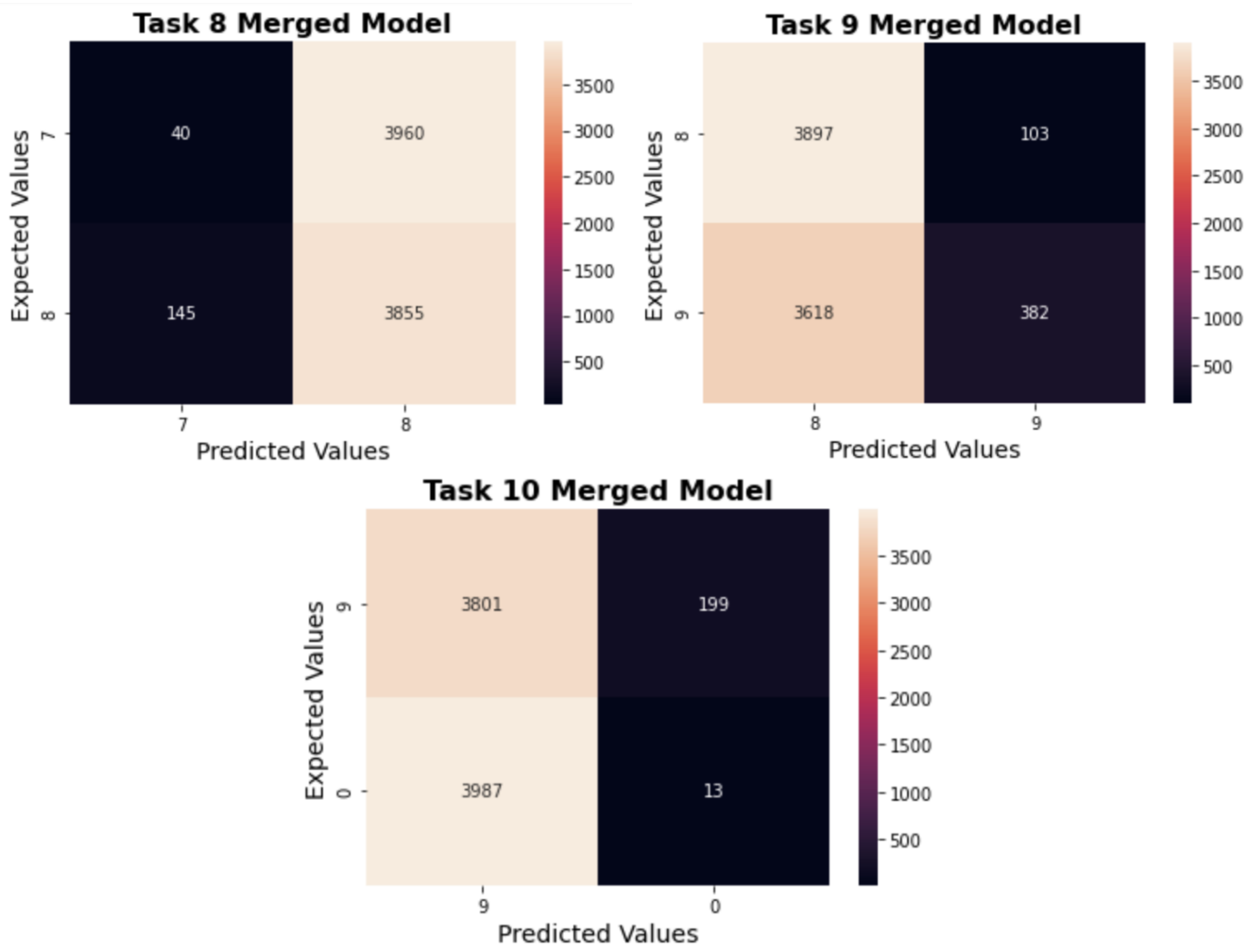


Figure 51: Confusion matrices of all 9 merged models

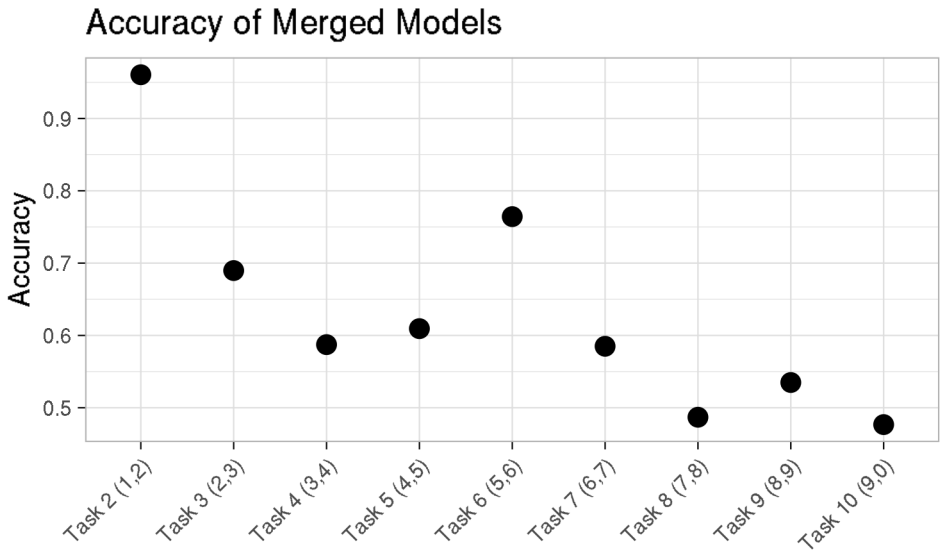


Figure 52: Average accuracies of all 9 merged models

As the last robustness check, the classifier layer of each merged model will be replaced with the classifier layers of all merged models as shown in Fig. 53. Unlike in the non-overlapped experiment, placing classifier layers of all merged models in overlapped experiment will yield 20 output nodes instead of 10. The question that arises then is how to classify digits 0 to 9? Recall that this experiment is implemented with an overlapped dataset. In other words, two output nodes are trained to classify the same number as in Fig. 53. To classify a digit between 0 and 9, the average value of the two output nodes that are trained to classify the same number are taken. After all average values are computed, these 10 values are fed through a SoftMax function to compare the probabilities

of the outputs. The prediction of the model will be taken as the index of the maximum of the 10 output probabilities.

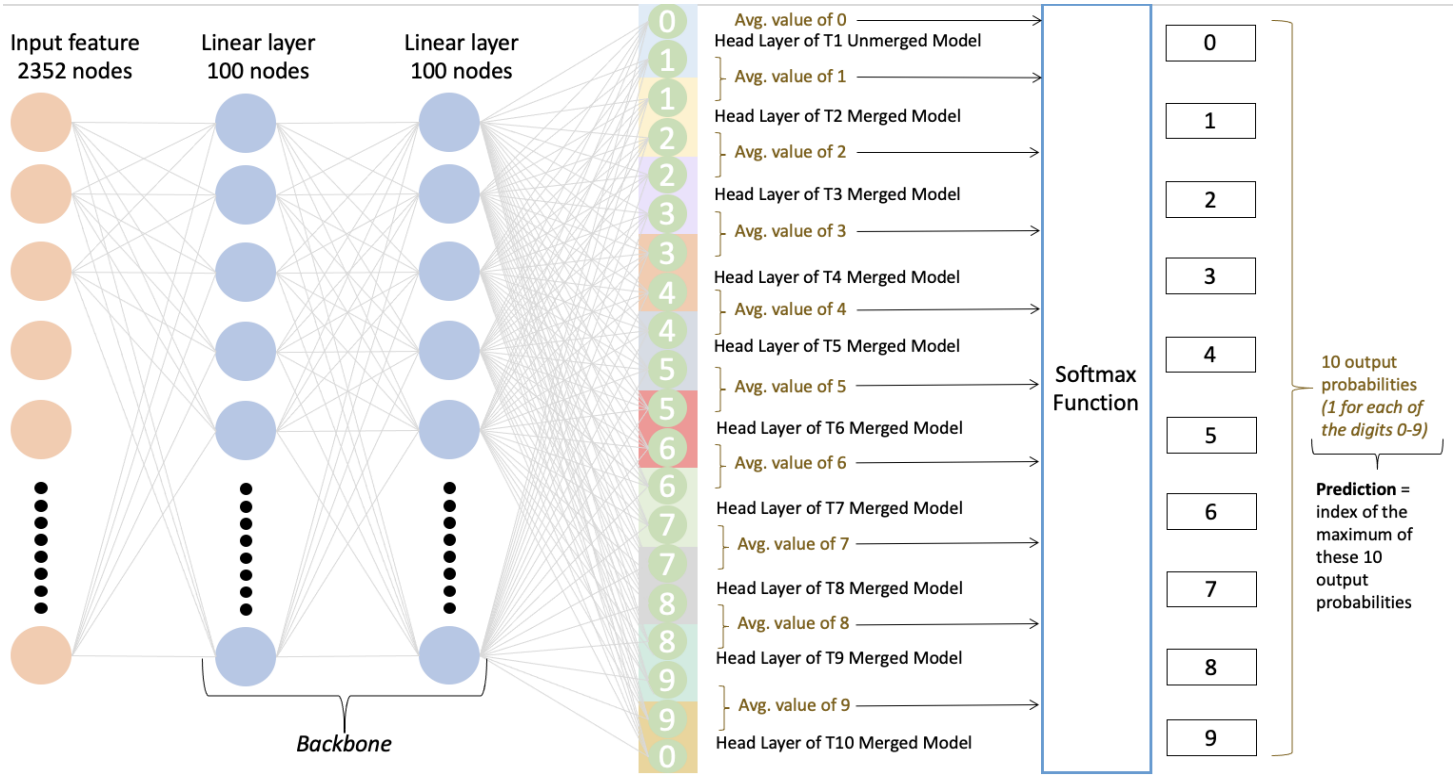
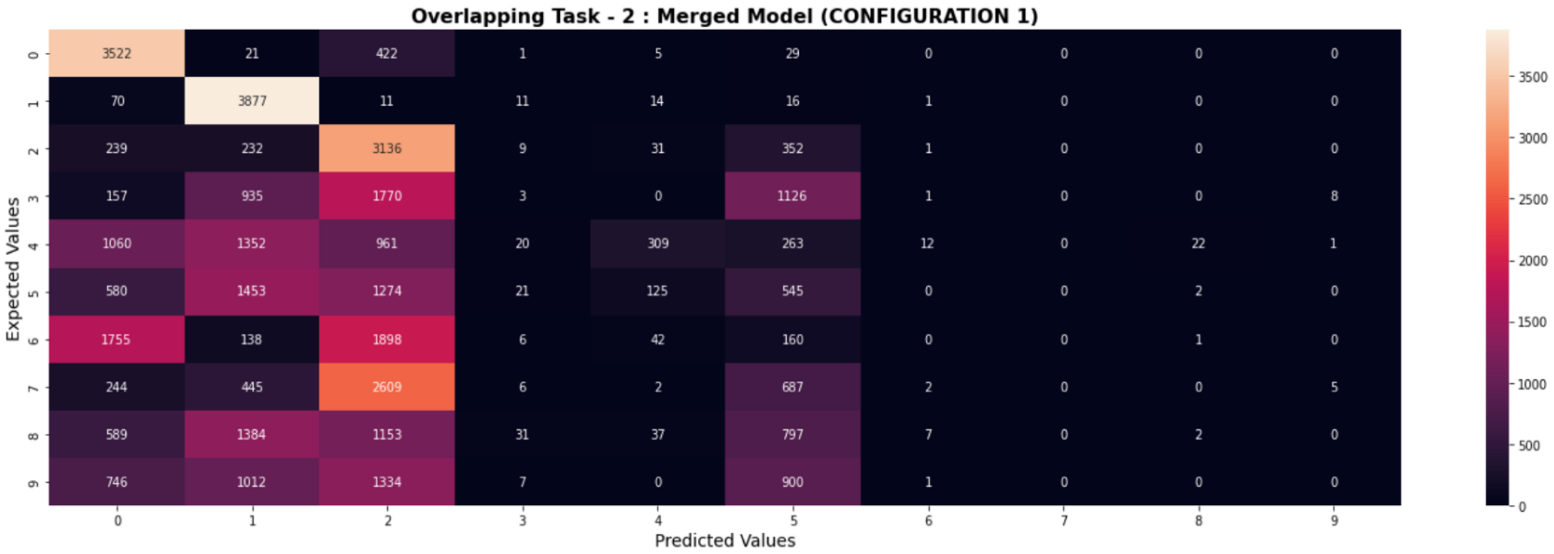
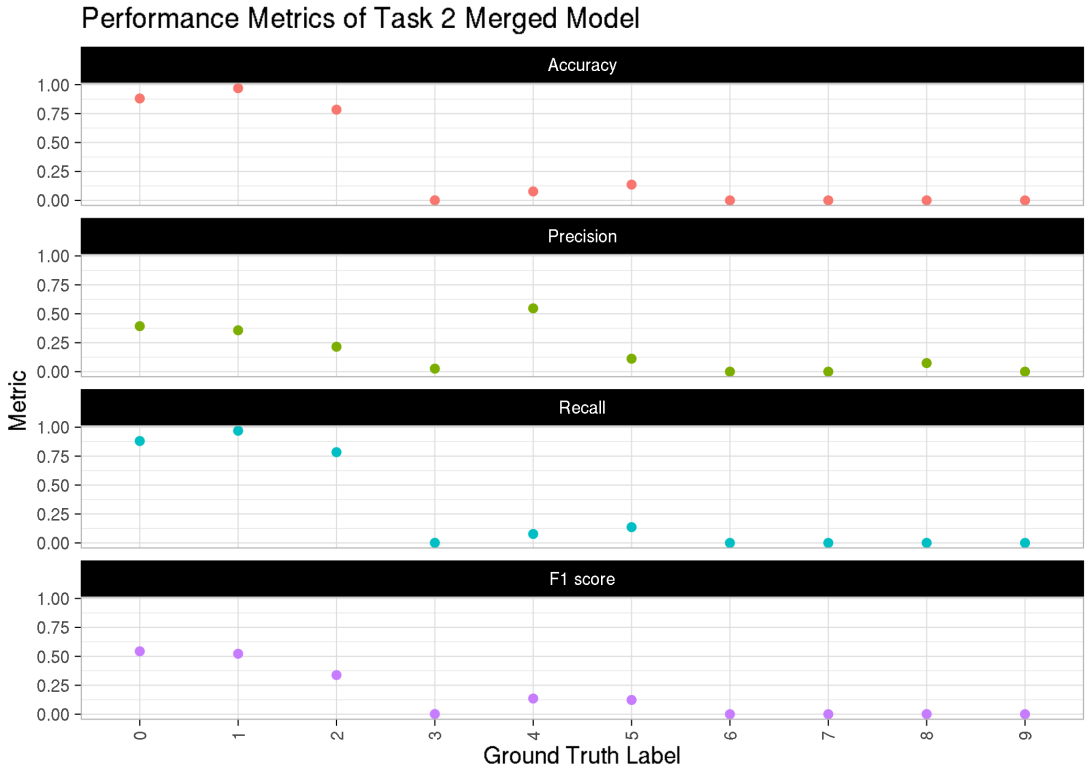


Figure 53: Model architecture of replaced classifier layer

The confusion matrix and performance metrics of task 2 merged model shown in Fig. 54(a) and Fig. 54(b), respectively. The confusion matrix and performance metrics of task 3 merged model shown in Fig. 55(a) and Fig. 55(b), respectively.



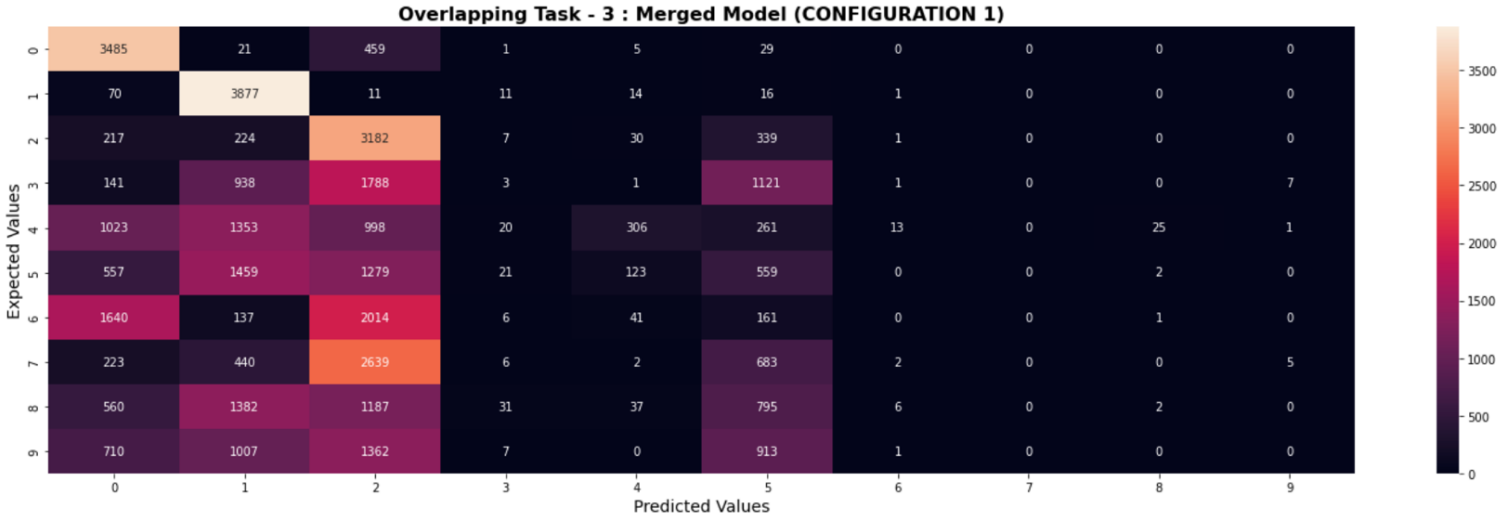
(a)



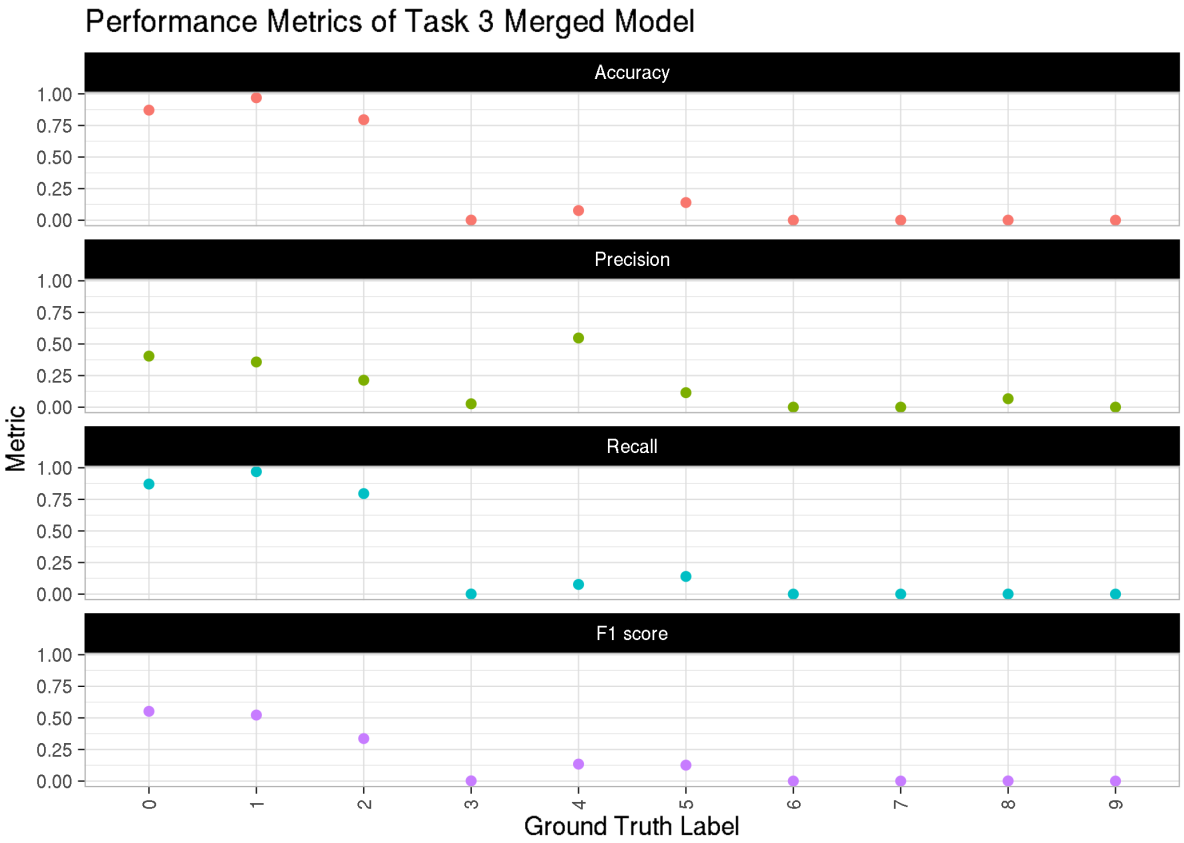
(b)

Figure 54: (a) Confusion Matrix and (b) Performance Metrics for Task 2 Merged Model

Based on the results shown in the confusion matrix and performance metrics of task 2 merged model, the model had high accuracies for predicting classes 0, 1 and 2. Among these three classifications, class 1 had the highest accuracy. The accuracy values for the rest of the classes were substantially low. This is expected behavior since the merged model of task 2 only contains the weighted average of both task 1 and task 2. The overall accuracy on all data is 0.2849.



(a)



(b)

Figure 55: (a) Confusion Matrix and (b) Performance Metrics for Task 3 Merged Model

The results for task 3 merged model are very similar to the results for task 2 merged model. For instance, this merged model also had high accuracies for predicting classes 0, 1 and 2. Among these three classifications, class 1 had the highest accuracy. There was a slight improvement in accuracy for identifying class 2 correctly. The accuracy values for the rest of the classes were substantially low. The slight improvement in identifying class 2 correctly could

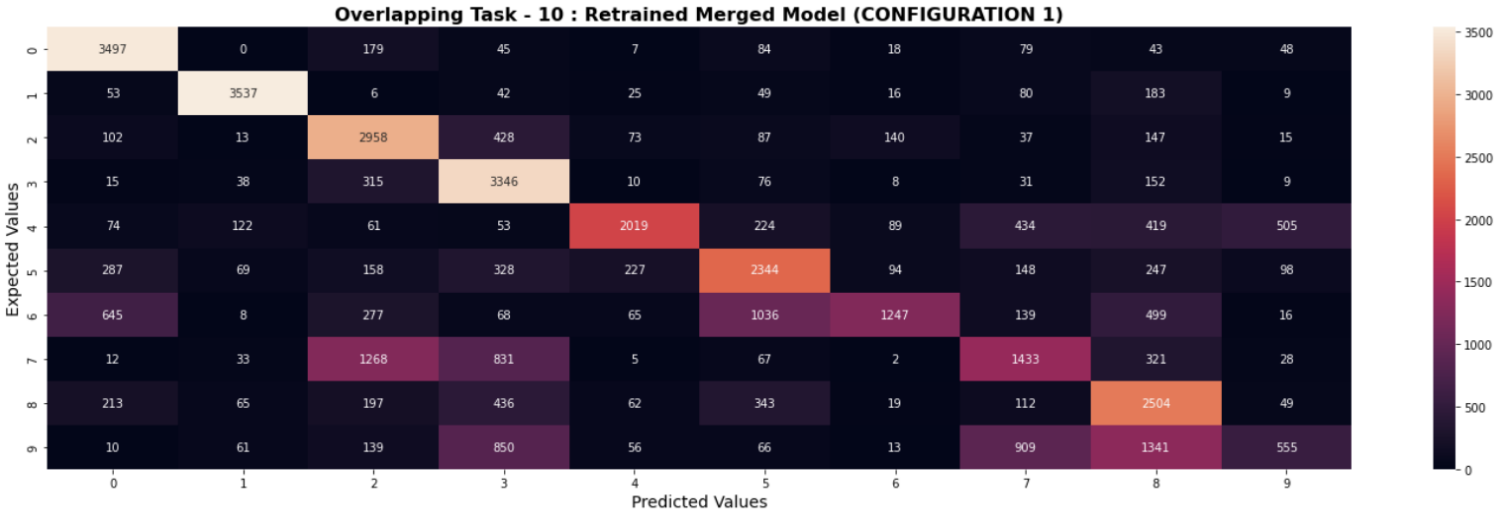
be because task 3 merged model contains weighted average of task 1, task 2, and task 3. The overall accuracy on all data is 0.2853.

The confusion matrices and performance of the rest of the merged models can be found in the supplementary section. After examining the results of each merged model closely, the overall accuracy on all data stays close to the same value.

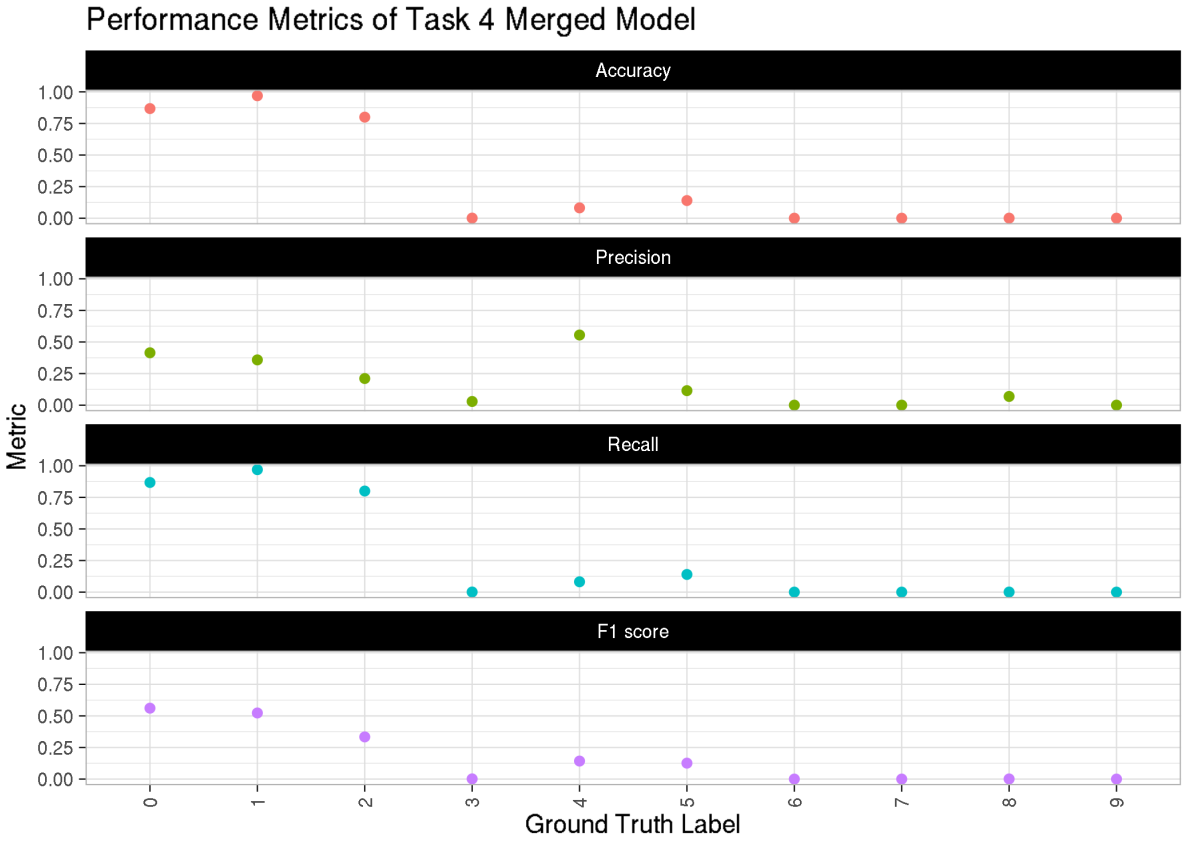
2) *Phase 2*

The task 10 merged model was used as the backbone of all the new model configurations.

a) *Model Configuration 1*: The parameters of the backbone architecture were kept frozen and only the head layer was trained. At the end of the execution, a model with the best validation accuracy of 0.6254 was received. Fig. 56 displays the confusion matrix and performance metrics of testing this model on EMNIST dataset. The overall accuracy is 0.5860.



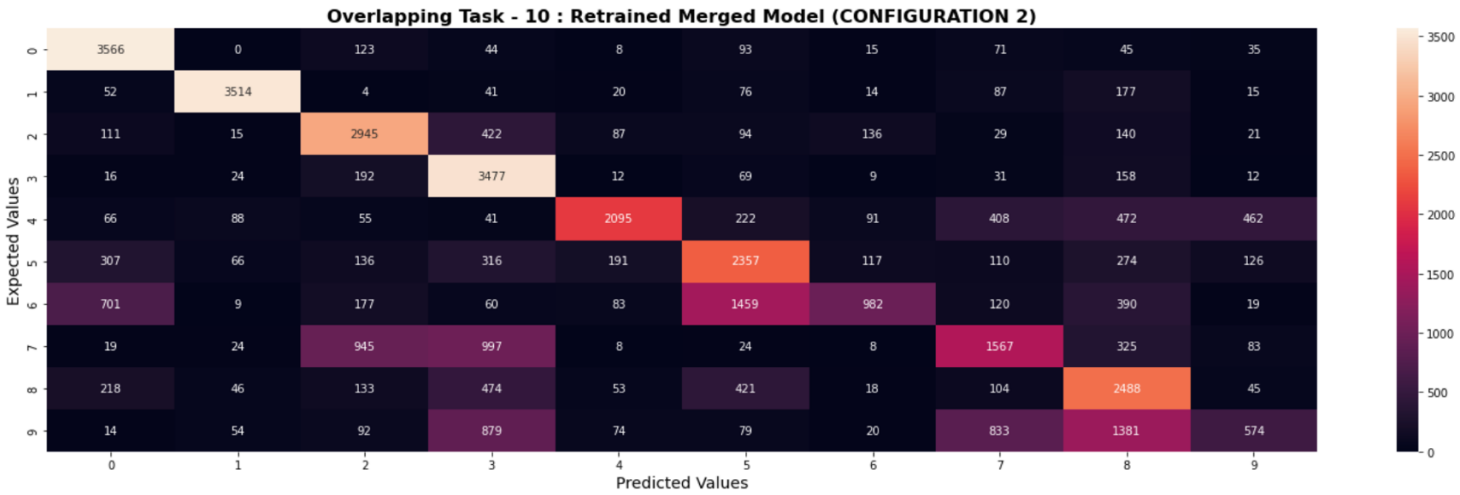
(a)



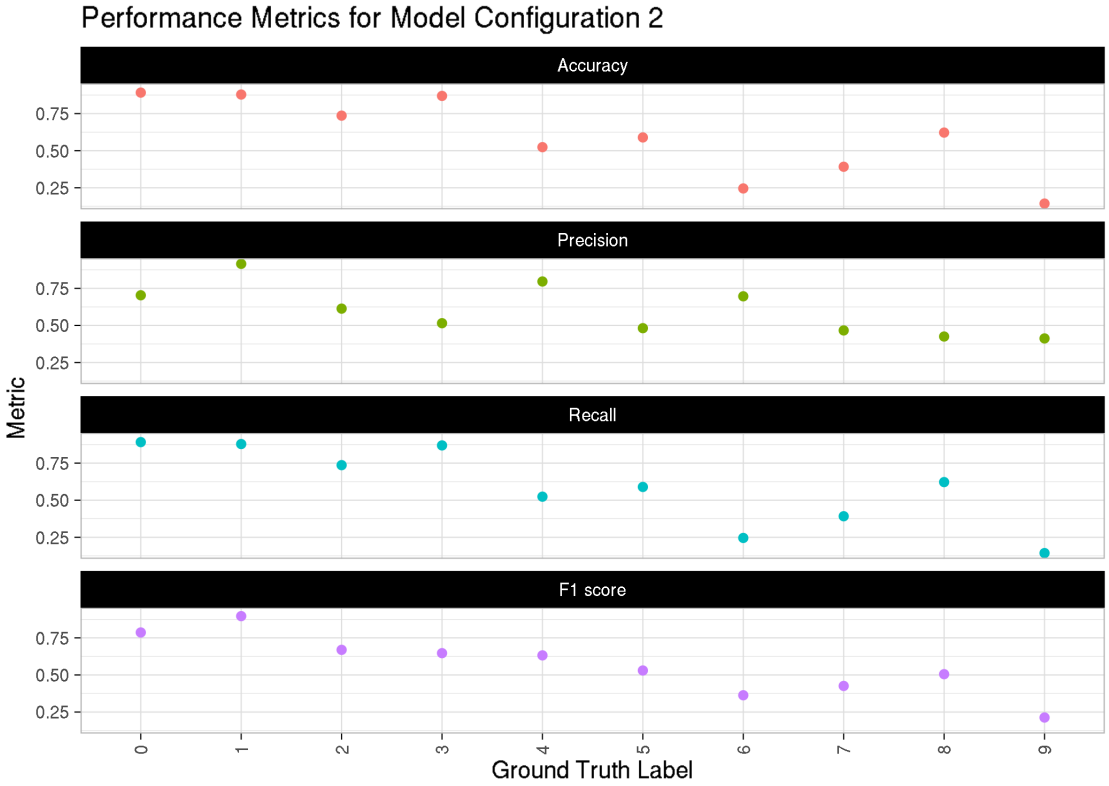
(b)

Figure 56: (a) Confusion Matrix and (b) Performance Metrics for Model Configuration 1 on entire EMNIST dataset

b) *Model Configuration 2:* The parameters of the backbone architecture were kept frozen and only the head layer was training. At the end of the execution, a model with the best validation accuracy of 0.6471 was received. Fig. 57 displays the confusion matrix and performance metrics of testing this model on EMNIST dataset. The overall accuracy on all data is 0.5891.



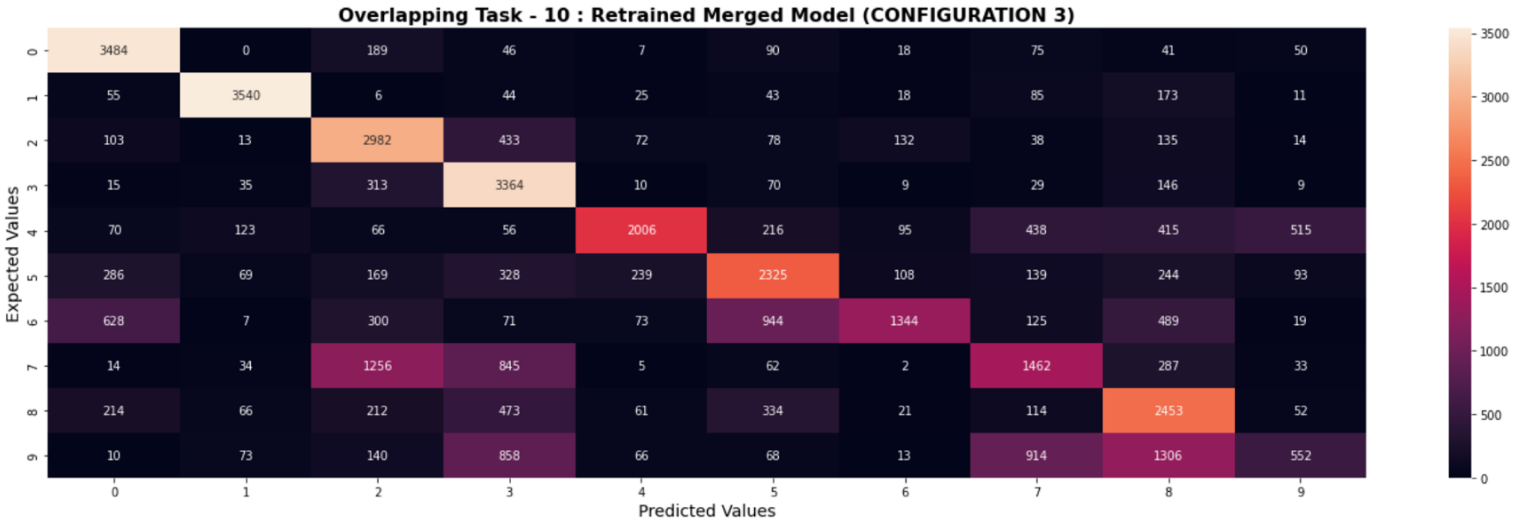
(a)



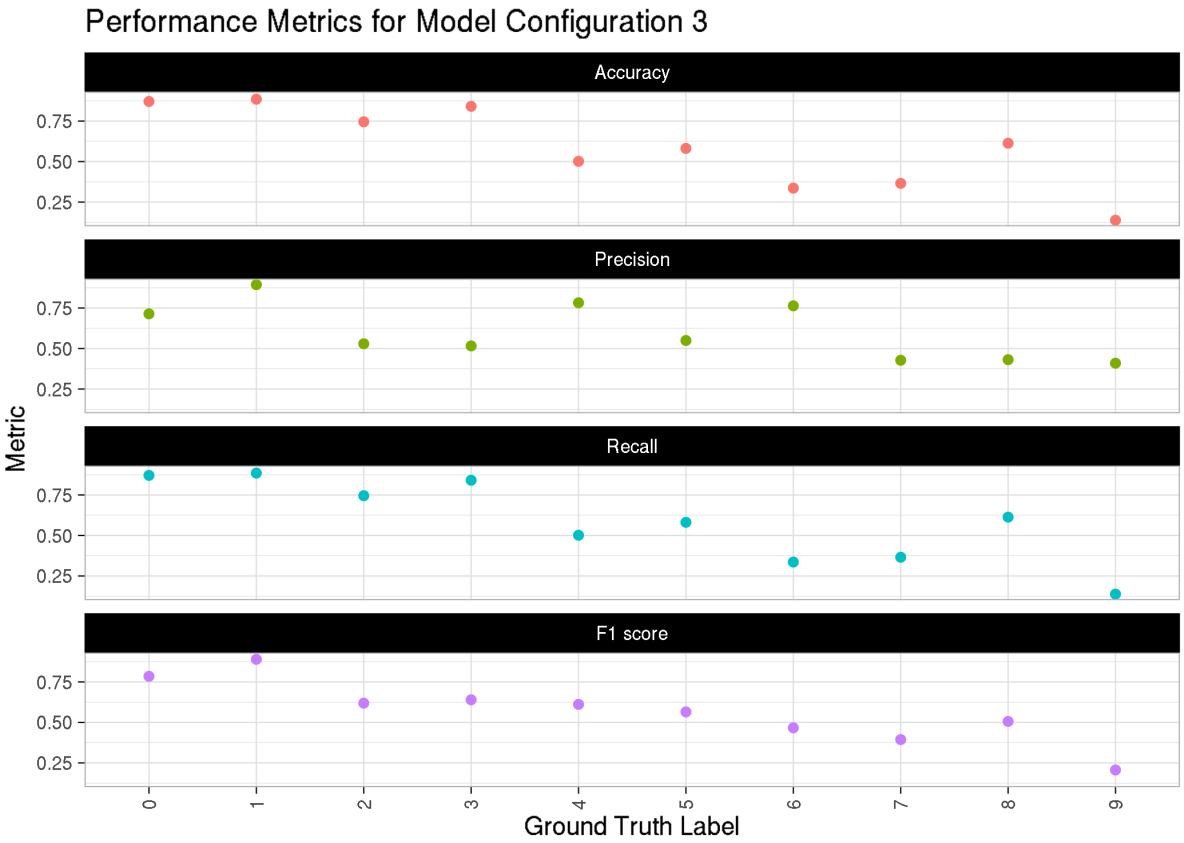
(b)

Figure 57: (a) Confusion Matrix and (b) Performance Metrics for Model Configuration 2 on entire EMNIST dataset

c) *Model Configuration 3:* This configuration was trained in the same manner as Configuration 1. At the end of the execution, a model with the best validation accuracy of 0.6247 was received. Fig. 58 displays the confusion matrix and performance metrics of testing this model on EMNIST dataset. The overall accuracy on all data is 0.5878.



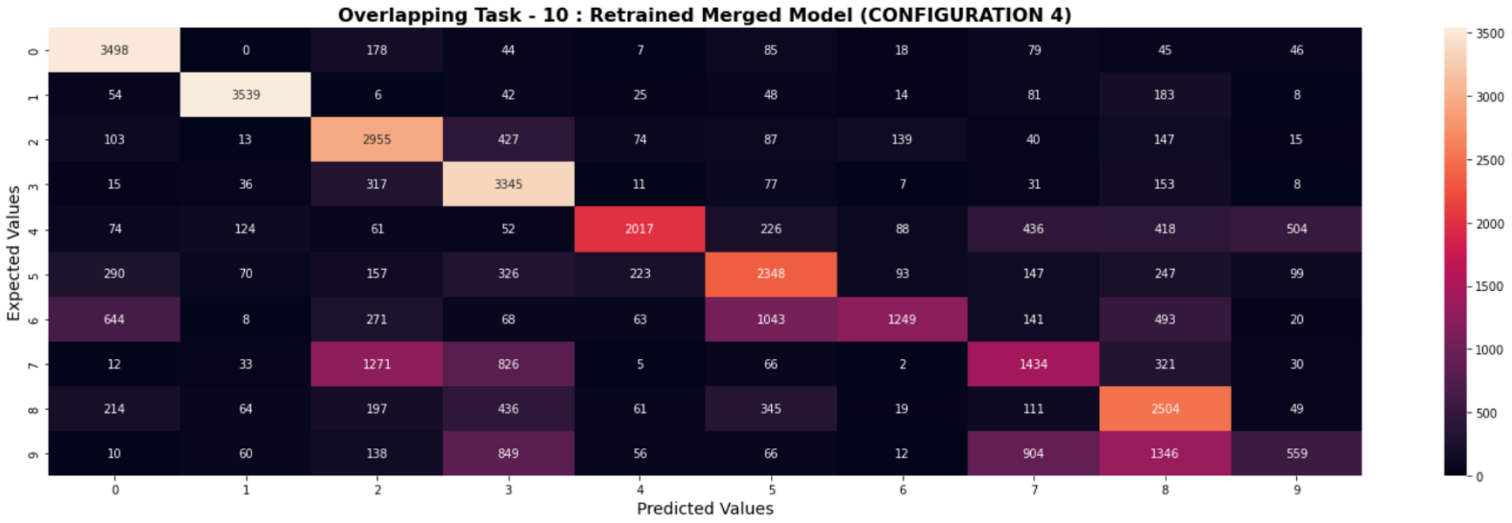
(a)



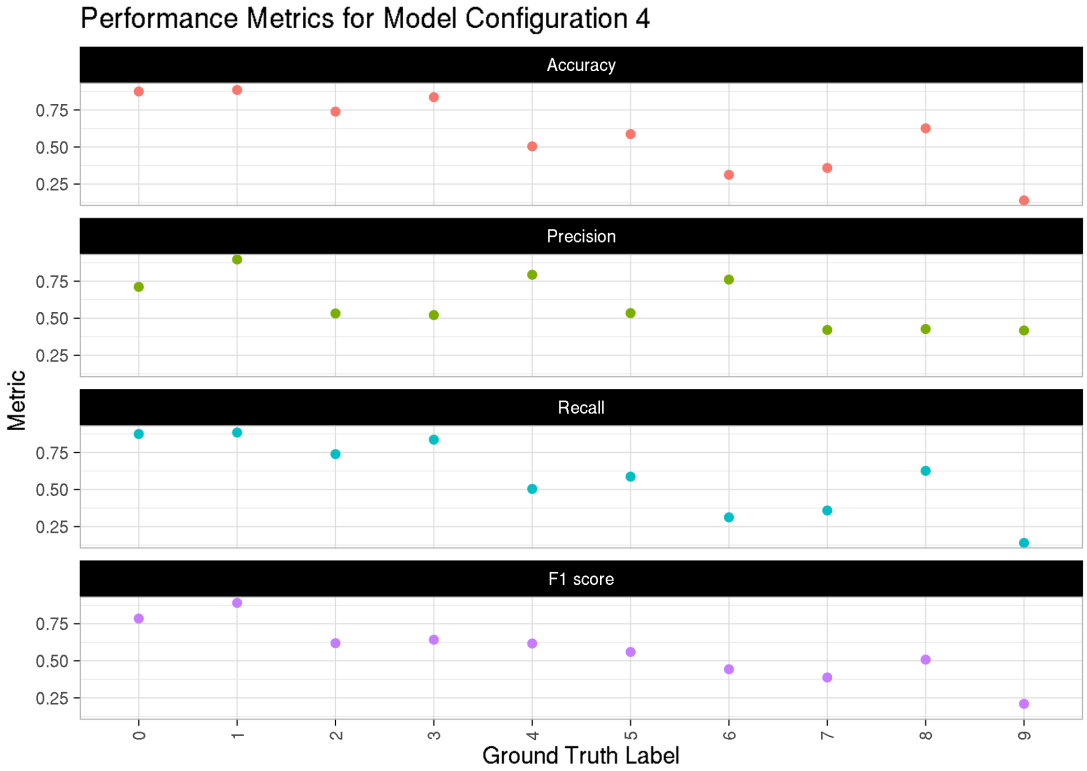
(b)

Figure 58: (a) Confusion Matrix and (b) Performance Metrics for Model Configuration 3 on entire EMNIST dataset

d) *Model Configuration 4:* After training the final layer while keeping the parameters of the backbone frozen, a model with the best validation accuracy of 0.6243 was retrieved. Fig. 59 displays the confusion matrix and performance metrics of testing this model on EMNIST dataset. The overall accuracy on all data is 0.5862.



(a)

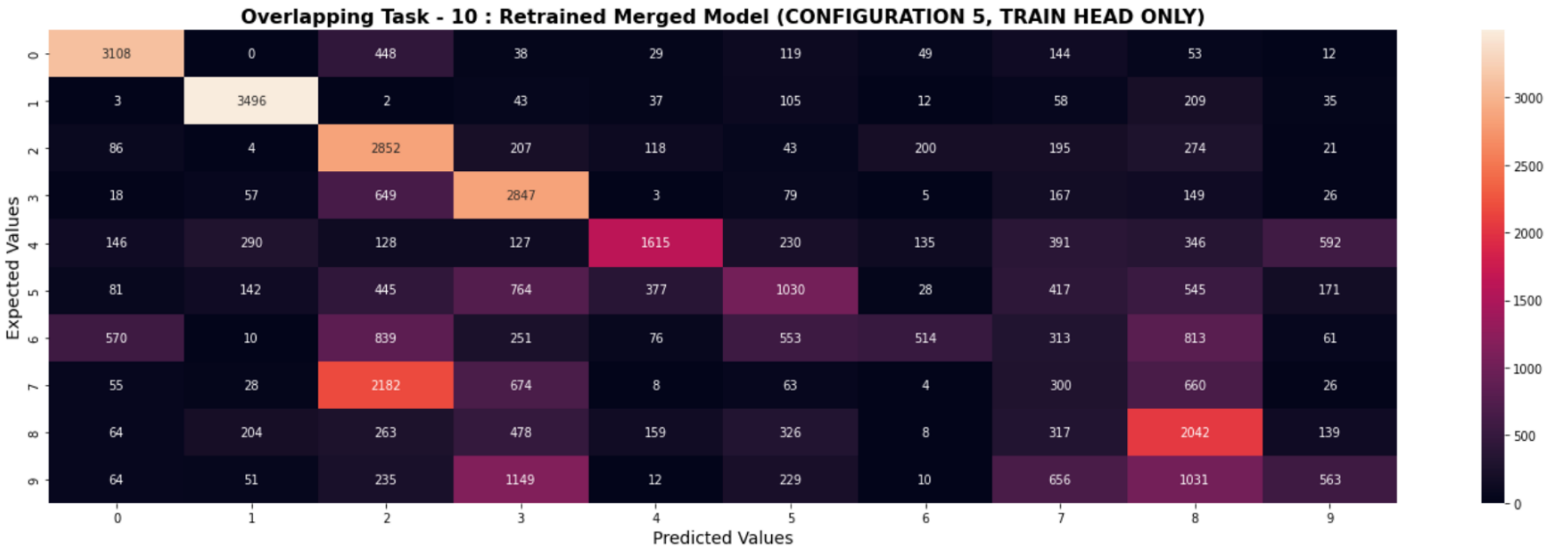


(b)

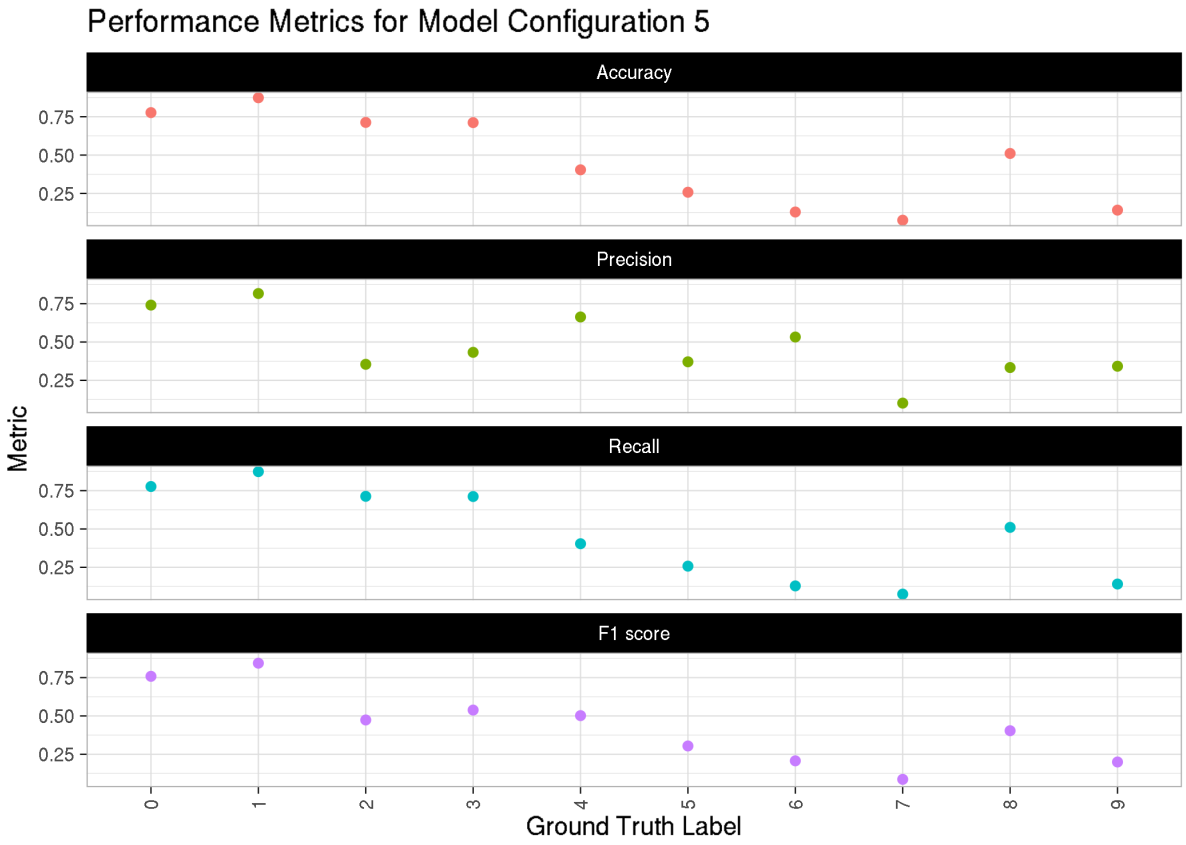
Figure 59: (a) Confusion Matrix and (b) Performance Metrics for Model Configuration 4 trained on head layer only on entire EMNIST dataset

e) *Model Configuration 5*: This configuration was trained in two different ways to see which training method would produce a good result.

i. In the first training method, the final layer was the only one that was trained while keeping the parameters of the backbone frozen. This resulted in a model with the best validation accuracy of 0.4846. Fig. 60 displays the confusion matrix and performance metrics of testing this model on EMNIST dataset. The overall accuracy on all data is 0.4592.



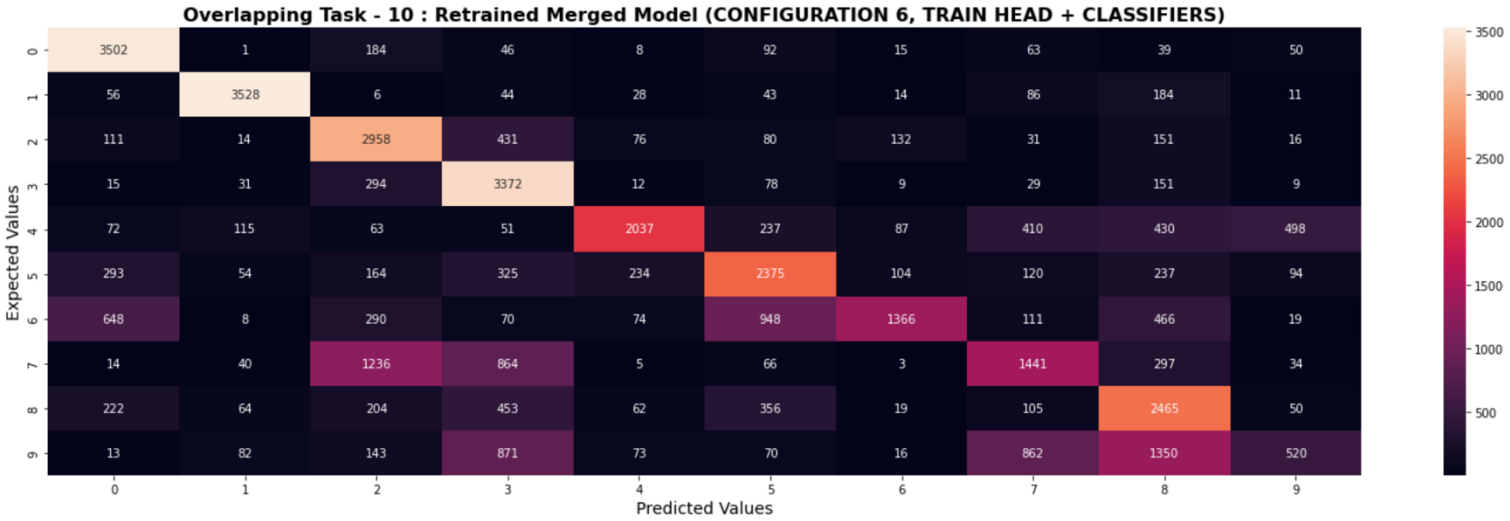
(a)



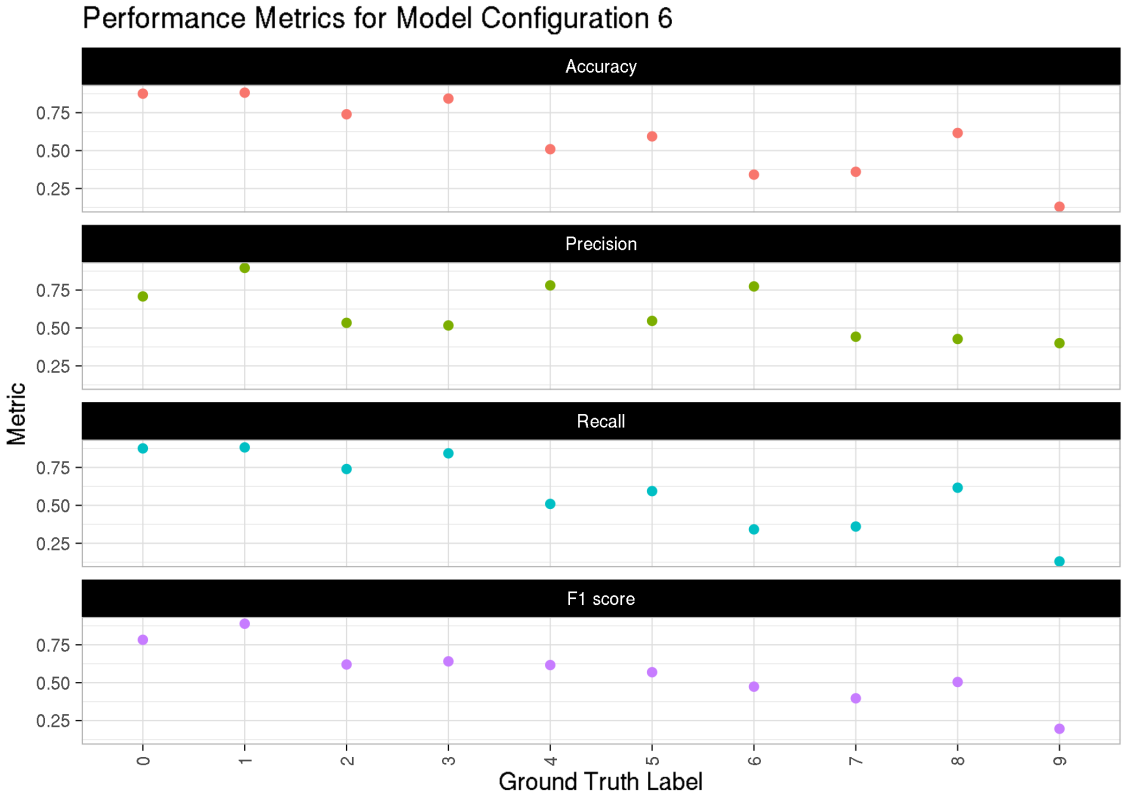
(b)

Figure 60: (a) Confusion Matrix and (b) Performance Metrics for Model Configuration 5 trained on head layer only on entire EMNIST dataset

ii. In the second training method, the head layer and final layer were the only two layers that were trained while keeping the parameters of the backbone frozen. This result in a model with the best validation accuracy of 0.6234. Fig. 61 displays the confusion matrix and performance metrics of testing this model on EMNIST dataset. The overall accuracy on all data is 0.5891.



(a)



(b)

Figure 61: (a) Confusion Matrix and (b) Performance Metrics for Model Configuration 6 trained on head layer only on entire EMNIST dataset

Based on the results received after training all 6 model configurations and testing each one on the entire EMNIST dataset, the accuracy provided by each model was very close to each other. Table XXIII showcases the results of the merged model of task 5, prior to training, on unseen task data. Table XXIV showcases the results of the merged model of task 5, whose last layer was retrained, on unseen data. Table XXV displays the percent change of performance metrics between non-retrained and retrained versions of task 5 merged model on unseen data. In this example, unseen task data was defined as {3,7} and it's taken from the QMNIST dataset.

TABLE XXIII: Performance Metrics for Merged Model of Task 5 – Non- Retrained

Category	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**TP**	**FP**	**TN**	**FN**
3	0.7408	0.92109	0.740	0.821191	3759	322	4881	1315
7	0	1.0	0	0	1	0	5074	5202

TABLE XXIV: Performance Metrics for Merged Model of Task 5 - Retrained

Category	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**TP**	**FP**	**TN**	**FN**
3	0.8638	0.9941	0.8638	0.924391	4383	26	5177	691
7	0.8647	0.9980	0.8647	0.926578	4499	9	5065	704

TABLE XXV: % Change in Performance Metrics between Non-retrained and Retrained (Task 5 Merged Model)

Category	**% Change in Accuracy**	**% Change in Precision**	**% Change in Recall**	**% Change in F1 Score**
3	16.6002	7.9259	16.6002	12.5672
7	inf	-0.1996	inf	inf

Table XXVI through Table XXVIII provide the change in performance metrics upon retraining for categories 3 and 7 for all 6 model configurations. In a real-world setting, user data cannot be predefined into a series of tasks so if the classifier layer of the final task model is retrained with all data categories, then the accuracy of identifying unseen data can be improved.

TABLE XXVI: Change in Performance Metrics Upon Retraining – Model Configuration 1

Category	**% Change in Accuracy**	**% Change in Precision**	**% Change in Recall**	**% Change in F1 Score**
3	40319.9987	298.6193	40319.9987	22540.5114
4	623.0768	33.81330	623.0768	339.2676

TABLE XXVII: Change in Performance Metrics Upon Retraining – Model Configuration 2

Category	**% Change in Accuracy**	**% Change in Precision**	**% Change in Recall**	**% Change in F1 Score**
3	67383.3324	1328.3944	67383.3324	38010.3157
4	1658.5903	44.0309	1658.5903	878.8898

TABLE XXVIII: Change in Performance Metrics Upon Retraining – Model Configuration 3

Category	**% Change in Accuracy**	**% Change in Precision**	**% Change in Recall**	**% Change in F1 Score**
3	40519.9983	298.9197	40519.9983	22610.2538
4	620.32967	34.1471	620.32967	338.9265

Similar to the previous experiment, we took another set of unseen data which we defined as {4, 7, 9} and evaluated how each model configuration worked against the merged model from the original experiment. Table XXIX showcases the results of the merged model of task 5, prior to training, on unseen task data. Table XXX showcases the results of the merged model of task 5, whose last layer was retrained, on unseen task data. Table XXXI displays the percent change of performance metrics between non-retrained and retrained versions of task 5 merged model on this unseen data.

TABLE XXIX: Performance Metrics for Merged Model of Task 5 – Non- Retrained

Category	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**TP**	**FP**	**TN**	**FN**
4	0	0	0	0	0	1	10028	4798
7	0.000192	1.0	0	0.000384	1	0	9624	5202
9	0.003937	0.3333	0.003	0.007782	19	38	9963	4807

TABLE XXX: Performance Metrics for Merged Model of Task 5 – Non- Retrained

Category	Accuracy	Precision	Recall	F1 Score	TP	FP	TN	FN
4	0.81679	0.8678	0.817	0.8415	3919	597	14742	879
7	0.8647	0.9394	0.865	0.9005	4499	290	9334	704
9	0.789266	0.8440	0.789	0.8157	3809	704	9297	1017

TABLE XXXI: % Change in Performance Metrics between Non-retrained and Retrained (Task 5 Merged Model)

Category	% Change in Accuracy	% Change in Precision	% Change in Recall	% Change in F1 Score
4	Inf	Inf	Inf	Inf
7	449800	-6.0555	449800	234215.4230
9	19947.3683	153.201	19947.3683	10381.9900

Table XXXII through Table XXXIV provide the change in performance metrics upon retraining for categories 4, 7 and 9 for all 6 model configurations.

TABLE XXXII: Change in Performance Metrics Upon Retraining – Model Configuration 1

Category	% Change in Accuracy	% Change in Precision	% Change in Recall	% Change in F1 Score
4	623.0768	33.81330	623.0768	339.2676
7	Inf	Inf	Inf	Inf
9	Inf	Inf	Inf	Inf

TABLE XXXIII: Change in Performance Metrics Upon Retraining – Model Configuration 2

Category	**% Change in Accuracy**	**% Change in Precision**	**% Change in Recall**	**% Change in F1 Score**
4	1658.5903	44.0309	1658.5903	878.8898
7	Inf	Inf	Inf	Inf
9	Inf	Inf	Inf	Inf

TABLE XXXIV: Change in Performance Metrics Upon Retraining – Model Configuration 3

Category	**% Change in Accuracy**	**% Change in Precision**	**% Change in Recall**	**% Change in F1 Score**
4	620.32967	34.1471	620.32967	338.9265
7	Inf	Inf	Inf	Inf
9	Inf	Inf	Inf	Inf

IX. CONCLUSION AND FUTURE WORK

This research project focused on the overarching idea of privacy-preserving visual recognition by closely examining federated learning frameworks, particularly the DUA framework. One of the challenges of this research project was understanding the DUA framework itself due to the different components it encompassed. Running a thorough investigation of the DUA framework, as part of the preliminary research, led to the discovery of a few pitfalls of the DUA framework. The primary pitfall that was found was this framework doesn't work on unseen data. As a brief recap, unseen data is defined as data that is on the user device that has not been used to train models on the server. Our research focused on addressing this primary pitfall by implementing an experiment that involved using overlapped datasets and developing multiple model

configurations and training them to determine which one performed the best on unseen data.

Based on our research, we found that nearly all 6 newly defined model configurations produced similar accuracy values on the entire EMNIST dataset, which we used as unseen data. Nonetheless, model configuration 2 provided the best results among all the model configurations because taking the maximum value of each category, or class, ensures that equal importance is given to all classes. For example, even though the accuracy value of model configuration was very close to that of model configuration 2, taking the average value of each class could sway the results especially if there is a large difference between the two values of each class.

A direct comparison of the original Numbers experiment and the experiment that we implemented suggests that we have produced a large overhead since the original experiments consisted of 5 models and our experiment produced 10 models. However, we have improved the accuracy by 1% on unseen data. In addition, we have also shown that we can improve the performance of unseen data by retraining the classifier layer of the final task merged model on all data categories.

This research is a stepping towards adapting DUA to other computer vision tasks. The future work of this research involves building DUA for other classification tasks such as human action recognition and object detection.

REFERENCES

[1] "Coronavirus creates boom in digital fitness - BBC News." https://www.bbc.com/news/technology-55318822 (accessed Sep. 07, 2021).

[2] T. Diethe, T. Borchert, E. Thereska, B. Balle, and N. Lawrence, "Continual Learning in Practice," Mar. 2019, Accessed: Sep. 07, 2021. [Online]. Available: http://arxiv.org/abs/1903.05202.

[3] N. Rieke *et al.*, "The future of digital health with federated learning," *npj Digit. Med. 2020 31*, vol. 3, no. 1, pp. 1–7, Sep. 2020, doi: 10.1038/s41746-020-00323-1.

[4] "U.S. data breaches by industry 2019 | Statista." https://www.statista.com/statistics/273572/number-of-data-breaches-in-the-united-states-by-business/ (accessed Sep. 07, 2021).

[5] "Google AI Blog: Federated Learning: Collaborative Machine Learning without Centralized Training Data." https://ai.googleblog.com/2017/04/federated-learning-collaborative.html (accessed Sep. 07, 2021).

[6] Q. Li *et al.*, "A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection," Jul. 2019, Accessed: Sep. 07, 2021. [Online]. Available: https://arxiv.org/abs/1907.09693.

[7] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," Feb. 2016, Accessed: Sep. 07, 2021. [Online]. Available: https://arxiv.org/abs/1602.05629.

[8] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," Oct. 2016, Accessed: Sep. 07, 2021. [Online]. Available: http://arxiv.org/abs/1610.05492.

[9] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic Federated Learning," Feb. 2019, Accessed: Sep. 07, 2021. [Online]. Available: https://arxiv.org/abs/1902.00146.

[10] Q. Li, Z. Wen, and B. He, "Practical Federated Gradient Boosting Decision Trees," Accessed: Sep. 07, 2021. [Online]. Available: www.aaai.org.

[11] S. Warnat-Herresthal *et al.*, "Swarm Learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, no. 7862, pp. 265–270, 2021, doi: 10.1038/s41586-021-03583-3.

[12] Y. Hu, Y. Zhou, J. Xiao, and C. Wu, "GFL: A Decentralized Federated Learning Framework Based On Blockchain," Oct. 2020, Accessed: Sep. 07, 2021. [Online]. Available: http://arxiv.org/abs/2010.10996.

[13] M. De Lange, X. Jia, S. Parisot, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "Unsupervised Model Personalization while Preserving Privacy and Scalability: An Open Problem," Mar. 2020, Accessed: Sep. 07, 2021. [Online]. Available: https://arxiv.org/abs/2003.13296.

[14] Z. Wu, Z. Wang, Z. Wang, and H. Jin, "Towards Privacy-Preserving Visual Recognition via Adversarial Training: A Pilot Study," Jul. 2018, Accessed: Sep. 07, 2021. [Online]. Available: https://arxiv.org/abs/1807.08379.

[15] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," 2018, [Online]. Available: http://arxiv.org/abs/1812.06127.

[16] Dwork C. (2006) Differential Privacy. In: Bugliesi M., Preneel B., Sassone V., Wegener I. (eds) Automata, Languages and Programming. ICALP 2006. Lecture Notes in Computer Science, vol 4052. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11787006_1

[17] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming Catastrophic Forgetting by Incremental Moment Matching," Mar. 2017, Accessed: Sep. 07, 2021. [Online]. Available: http://arxiv.org/abs/1703.08475.

[18] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, Y. Khazaeni, "Federated Learning with Matched Averaging," 2020, Available: https://openreview.net/forum?id=BkluqlSFDS

[19] Y. LeCun, C. Cortes, C. J. C. Burges, "The MNIST database of handwritten digits," Available: yann.lecun.com/exdb/mnist

[20] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, "Reading Digits in Natural Languages with Unsupervised Feature Learning," 2011. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning.* Available: http://ufldl.stanford.edu/housenumbers/

[21] A. Quattoni, A. Torralba. "Recognizing Indoor Scene," *IEEE Conference on Computer Vision and Pattern Recognition 2009.*

[22] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, T. Tuytelaars, "Memory Aware Synapses: Learning what (not) to forget," Oct. 5, 2018. *ECCV 2018*. Available: https://arxiv.org/abs/1711.09601

[23] R. Johari, "Lecture 16: Bayesian Inference," Available: http://web.stanford.edu/~rjohari/teaching/notes/226_lecture16_inference.pdf

[24] K. K. Gan "L3: Gaussian Probability Distribution," Available: https://www.asc.ohio-state.edu/gan.1/teaching/spring04/Chapter3.pdf

[25] G. Oren, L. Wolf, "In Defense of the Learning Without Forgetting for Task Incremental Learning," *CVPR 2021.* Available: https://arxiv.org/abs/2107.12304

[26] S. Jha. "Continual Learning – Where Are We?," Available: https://towardsdatascience.com/continual-learning-where-are-we-d5706e78a295

[27] A. Ly, M. Marsman, J. Verhagen, R. Grasman, E. Wagenmakers, "A Tutorial on Fisher Information," Oct. 17, 2017. Available: https://arxiv.org/pdf/1705.01064.pdf

[28] B. Yao, X. Jiang, A. Khosla, A.L. Lin, L.J. Guibas, and L. Fei-Fei,"Human Action Recognition by Learning Bases of Action Attributes and Parts." *Internation Conference on Computer Vision (ICCV)*, Barcelona, Spain. Nov. 6-13, 2011.

[29] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, K. Talwar, "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data." *ICLR 2017.* Available: https://arxiv.org/pdf/1610.05755.pdf

[30] Y. Zhu, X. Yu, M. Chandraker, Y. Wang, "Private-kNN: Practical Differential Privacy for Computer Vision," *CVPR 2020.* Available: https://ieeexplore.ieee.org/document/9156598

SUPPLEMENTARY

The tables below showcase the performance metrics of the unmerged and merged models of the MIT Indoor Scenes experiment as part of the robustness check that was done using a custom scenes dataset. The process of how the evaluation was performed can be found in Section X. The accuracy values for each task evaluation were 0.5255, 0.5092, 0.5824 and 0.5303 for all unmerged models, respectively. The accuracy values for each task evaluation for all merged models was 0.5053, 0.4915 and 0.4797, respectively.

TABLE SI: Performance Metrics for Task 1 Unmerged Model

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	bathroom	0.695652	0.659794	0.695652	0.677249	64	33	1150	28	unmerged_T1
1	bedroom	0.318681	0.604167	0.318681	0.417266	29	19	1165	62	unmerged_T1
2	children_room	0.241758	0.305556	0.241758	0.269939	22	50	1134	69	unmerged_T1
3	closet	0.517647	0.771930	0.517647	0.619718	44	13	1177	41	unmerged_T1
4	corridor	0.652174	0.666667	0.652174	0.659341	60	30	1153	32	unmerged_T1
5	dining_room	0.473118	0.771930	0.473118	0.586667	44	13	1169	49	unmerged_T1
6	garage	0.404255	0.422222	0.404255	0.413043	38	52	1129	56	unmerged_T1
7	kitchen	0.777778	0.496454	0.777778	0.606061	70	71	1114	20	unmerged_T1
8	livingroom	0.634409	0.373418	0.634409	0.470120	59	99	1083	34	unmerged_T1
9	lobby	0.611111	0.323529	0.611111	0.423077	55	115	1070	35	unmerged_T1
10	nursery	0.073684	0.304348	0.073684	0.118644	7	16	1164	88	unmerged_T1
11	pantry	0.366667	0.825000	0.366667	0.507692	33	7	1178	57	unmerged_T1
12	stairscase	0.879121	0.601504	0.879121	0.714286	80	53	1131	11	unmerged_T1
13	winecellar	0.738636	0.656566	0.738636	0.695187	65	34	1153	23	unmerged_T1

TABLE S2: Performance Metrics for Task 2 Unmerged Model

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	bar	0.410526	0.481481	0.410526	0.443182	39	42	892	56	unmerged_T2
1	bowling	0.711340	0.492857	0.711340	0.582278	69	71	861	28	unmerged_T2
2	buffet	0.287234	0.729730	0.287234	0.412214	27	10	925	67	unmerged_T2
3	casino	0.404494	0.493151	0.404494	0.444444	36	37	903	53	unmerged_T2
4	concert_hall	0.872340	0.458101	0.872340	0.600733	82	97	838	12	unmerged_T2
5	fastfood_restaurant	0.677419	0.440559	0.677419	0.533898	63	80	856	30	unmerged_T2
6	gameroom	0.318681	0.537037	0.318681	0.400000	29	25	913	62	unmerged_T2
7	gym	0.557895	0.768116	0.557895	0.646342	53	16	918	42	unmerged_T2
8	hairsalon	0.461538	0.583333	0.461538	0.515337	42	30	908	49	unmerged_T2
9	movietheater	0.583333	0.565657	0.583333	0.574359	56	43	890	40	unmerged_T2
10	restaurant	0.297872	0.341463	0.297872	0.318182	28	54	881	66	unmerged_T2

TABLE S3: Performance Metrics for Task 3 Unmerged Model

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	airport_inside	0.597826	0.384615	0.597826	0.468085	55	88	1118	37	unmerged_T3
1	church_inside	0.858696	0.593985	0.858696	0.702222	79	54	1152	13	unmerged_T3
2	cloister	0.755556	0.772727	0.755556	0.764045	68	20	1188	22	unmerged_T3
3	elevator	0.351064	0.458333	0.351064	0.397590	33	39	1165	61	unmerged_T3
4	inside_bus	0.705263	0.761364	0.705263	0.732240	67	21	1182	28	unmerged_T3
5	inside_subway	0.666667	0.646465	0.666667	0.656410	64	35	1167	32	unmerged_T3
6	library	0.649485	0.840000	0.649485	0.732558	63	12	1189	34	unmerged_T3
7	locker_room	0.445652	0.518987	0.445652	0.479532	41	38	1168	51	unmerged_T3
8	museum	0.393617	0.462500	0.393617	0.425287	37	43	1161	57	unmerged_T3
9	poolinside	0.685393	0.734940	0.685393	0.709302	61	22	1187	28	unmerged_T3
10	prisoncell	0.510870	0.652778	0.510870	0.573171	47	25	1181	45	unmerged_T3
11	subway	0.478723	0.432692	0.478723	0.454545	45	59	1145	49	unmerged_T3
12	trainstation	0.439560	0.421053	0.439560	0.430108	40	55	1152	51	unmerged_T3
13	waitingroom	0.622222	0.643678	0.622222	0.632768	56	31	1177	34	unmerged_T3

TABLE S4: Performance Metrics for Task 4 Unmerged Model

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	bakery	0.393617	0.513889	0.393617	0.445783	37	35	978	57	unmerged_T4
1	bookstore	0.474227	0.707692	0.474227	0.567901	46	19	991	51	unmerged_T4
2	clothingstore	0.626374	0.542857	0.626374	0.581633	57	48	968	34	unmerged_T4
3	deli	0.136842	0.254902	0.136842	0.178082	13	38	974	82	unmerged_T4
4	florist	0.755814	0.915493	0.755814	0.828025	65	6	1015	21	unmerged_T4
5	grocerystore	0.627660	0.508621	0.627660	0.561905	59	57	956	35	unmerged_T4
6	jewelleryshop	0.437500	0.287671	0.437500	0.347107	42	104	907	54	unmerged_T4
7	laundromat	0.839080	0.901235	0.839080	0.869048	73	8	1012	14	unmerged_T4
8	mall	0.836957	0.531034	0.836957	0.649789	77	68	947	15	unmerged_T4
9	shoeshop	0.595745	0.565657	0.595745	0.580311	56	43	970	38	unmerged_T4
10	toystore	0.531250	0.472222	0.531250	0.500000	51	57	954	45	unmerged_T4
11	videostore	0.129412	0.229167	0.129412	0.165414	11	37	985	74	unmerged_T4

TABLE S5: Performance Metrics for Task 2 Merged Model

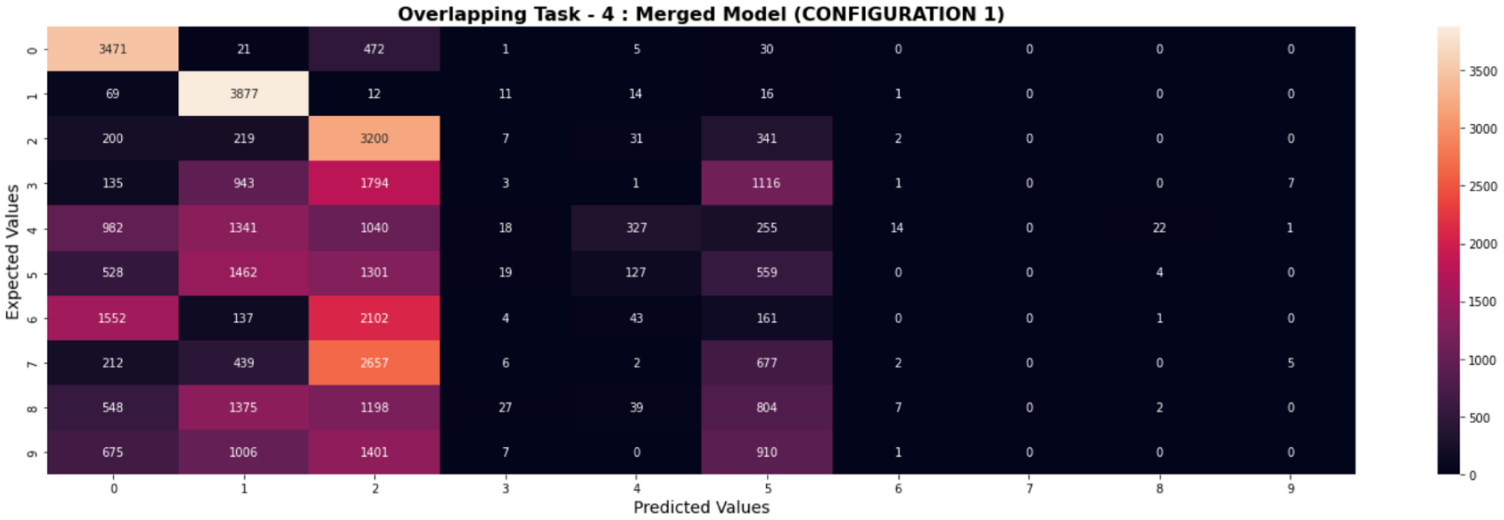
	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	bar	0.305263	0.439394	0.305263	0.360248	29	37	897	66	merged_T2
1	bowling	0.711340	0.526718	0.711340	0.605263	69	62	870	28	merged_T2
2	buffet	0.287234	0.729730	0.287234	0.412214	27	10	925	67	merged_T2
3	casino	0.460674	0.445652	0.460674	0.453039	41	51	889	48	merged_T2
4	concert_hall	0.829787	0.493671	0.829787	0.619048	78	80	855	16	merged_T2
5	fastfood_restaurant	0.731183	0.412121	0.731183	0.527132	68	97	839	25	merged_T2
6	gameroom	0.296703	0.562500	0.296703	0.388489	27	21	917	64	merged_T2
7	gym	0.547368	0.800000	0.547368	0.650000	52	13	921	43	merged_T2
8	hairsalon	0.428571	0.582090	0.428571	0.493671	39	28	910	52	merged_T2
9	movietheater	0.614583	0.526786	0.614583	0.567308	59	53	880	37	merged_T2
10	restaurant	0.329787	0.352273	0.329787	0.340659	31	57	878	63	merged_T2

TABLE S6: Performance Metrics for Task 3 Merged Model

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	airport_inside	0.445652	0.303704	0.445652	0.361233	41	94	1112	51	merged_T3
1	church_inside	0.826087	0.589147	0.826087	0.687783	76	53	1153	16	merged_T3
2	cloister	0.777778	0.804598	0.777778	0.790960	70	17	1191	20	merged_T3
3	elevator	0.297872	0.509091	0.297872	0.375839	28	27	1177	66	merged_T3
4	inside_bus	0.326316	0.885714	0.326316	0.476923	31	4	1199	64	merged_T3
5	inside_subway	0.552083	0.464912	0.552083	0.504762	53	61	1141	43	merged_T3
6	library	0.494845	0.888889	0.494845	0.635762	48	6	1195	49	merged_T3
7	locker_room	0.554348	0.320755	0.554348	0.406374	51	108	1098	41	merged_T3
8	museum	0.287234	0.415385	0.287234	0.339623	27	38	1166	67	merged_T3
9	poolinside	0.404494	0.923077	0.404494	0.562500	36	3	1206	53	merged_T3
10	prisoncell	0.391304	0.666667	0.391304	0.493151	36	18	1188	56	merged_T3
11	subway	0.638298	0.314136	0.638298	0.421053	60	131	1073	34	merged_T3
12	trainstation	0.428571	0.364486	0.428571	0.393939	39	68	1139	52	merged_T3
13	waitingroom	0.466667	0.567568	0.466667	0.512195	42	32	1176	48	merged_T3

TABLE S7: Performance Metrics for Task 4 Merged Model

	category	accuracy	precision	recall	F1 score	TP	FP	TN	FN	model
0	bakery	0.617021	0.355828	0.617021	0.451362	58	105	908	36	merged_T4
1	bookstore	0.474227	0.541176	0.474227	0.505495	46	39	971	51	merged_T4
2	clothingstore	0.494505	0.542169	0.494505	0.517241	45	38	978	46	merged_T4
3	deli	0.094737	0.214286	0.094737	0.131387	9	33	979	86	merged_T4
4	florist	0.581395	0.980392	0.581395	0.729927	50	1	1020	36	merged_T4
5	grocerystore	0.595745	0.430769	0.595745	0.500000	56	74	939	38	merged_T4
6	jewelleryshop	0.312500	0.315789	0.312500	0.314136	30	65	946	66	merged_T4
7	laundromat	0.873563	0.926829	0.873563	0.899408	76	6	1014	11	merged_T4
8	mall	0.619565	0.600000	0.619565	0.609626	57	38	977	35	merged_T4
9	shoeshop	0.627660	0.366460	0.627660	0.462745	59	102	911	35	merged_T4
10	toystore	0.375000	0.514286	0.375000	0.433735	36	34	977	60	merged_T4
11	videostore	0.105882	0.180000	0.105882	0.133333	9	41	981	76	merged_T4



(a)

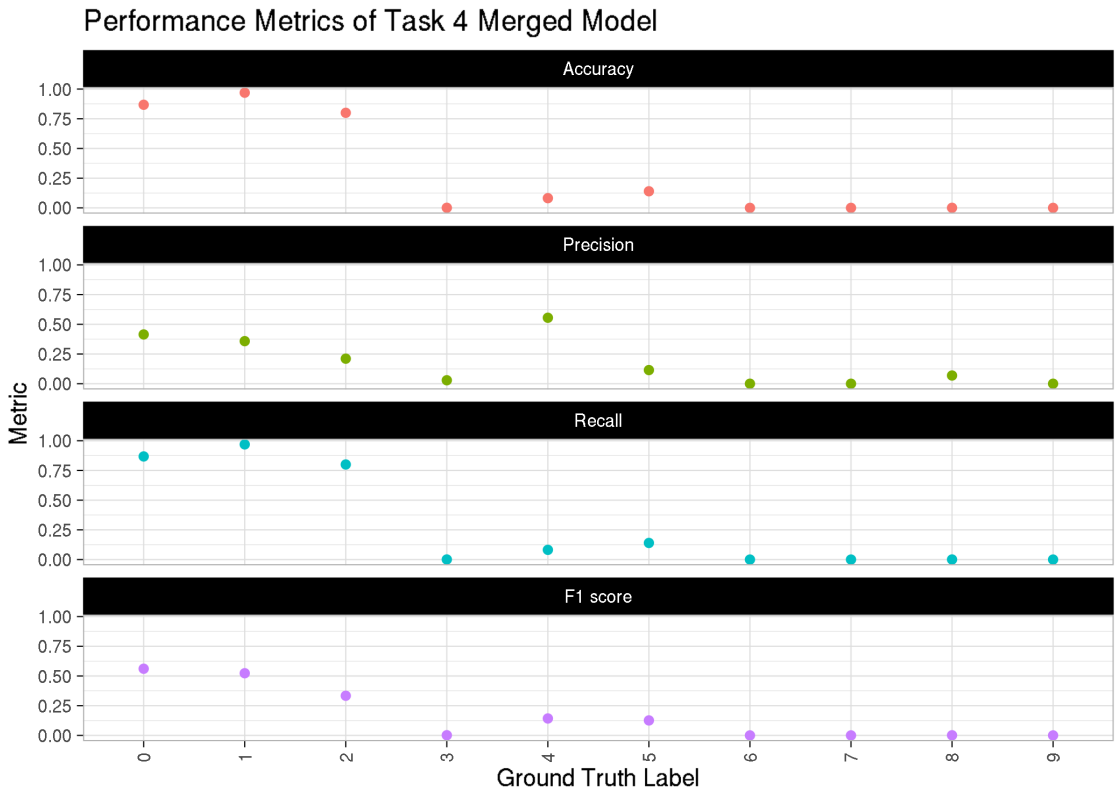
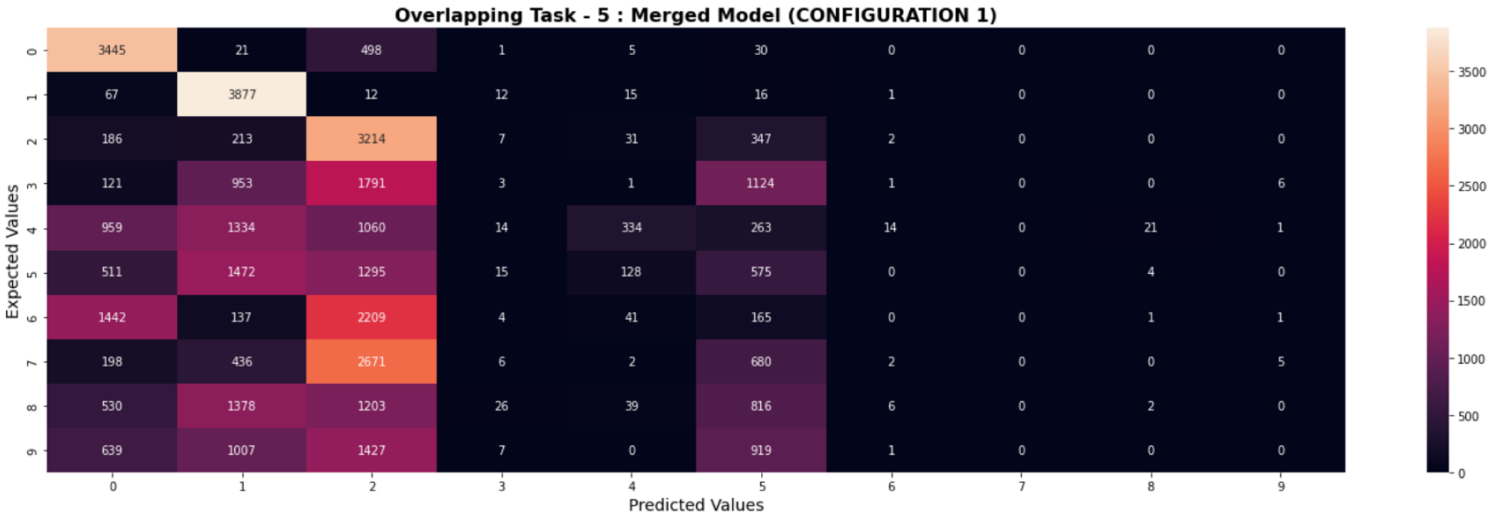
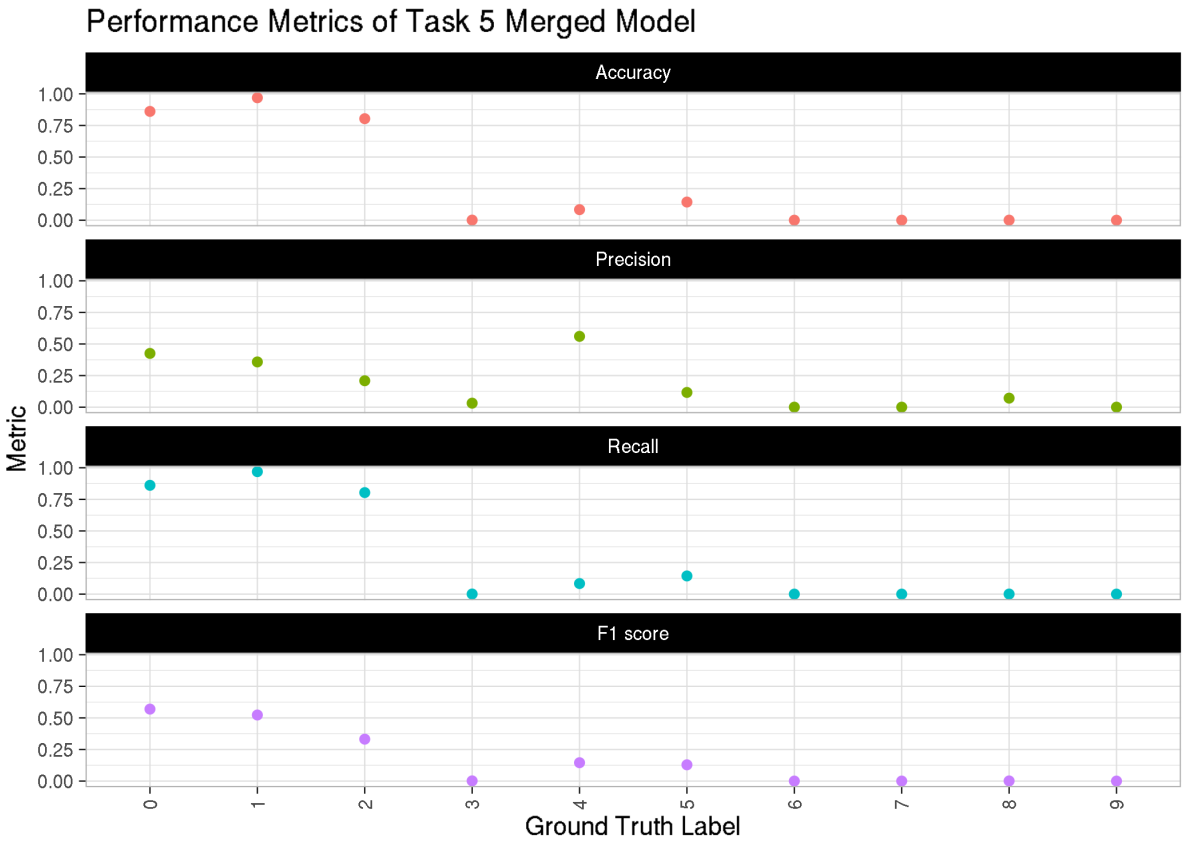


Figure S1: (a) Confusion Matrix and Performance Metrics for Task 4 Merged Model on entire EMNIST dataset

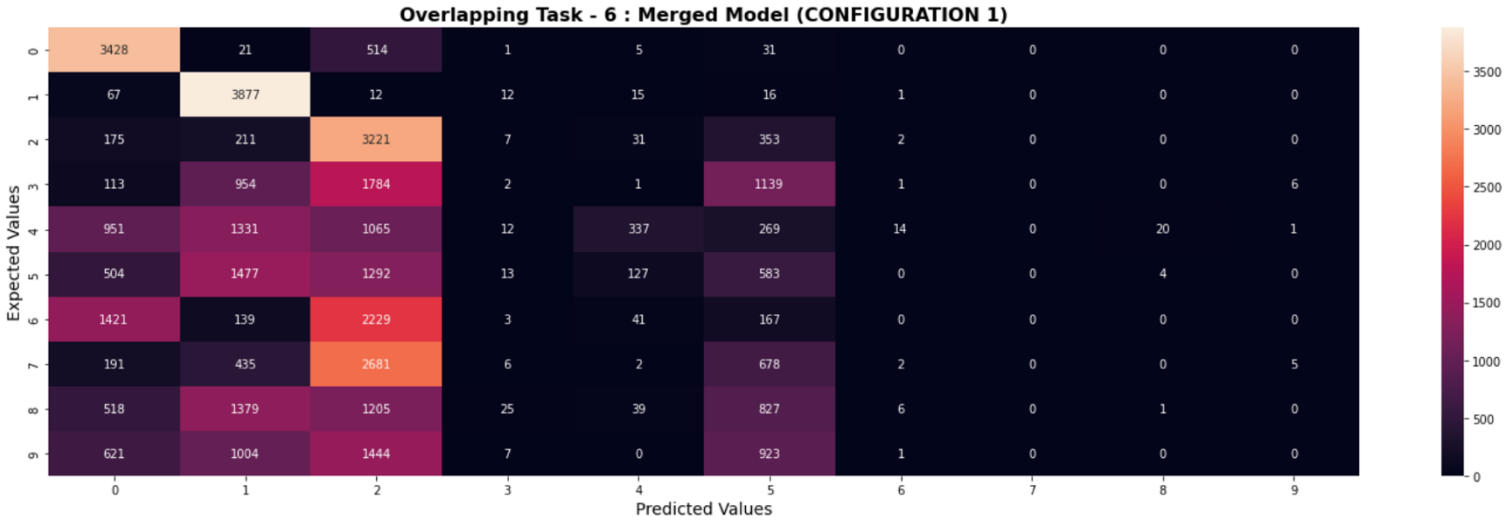


(a)

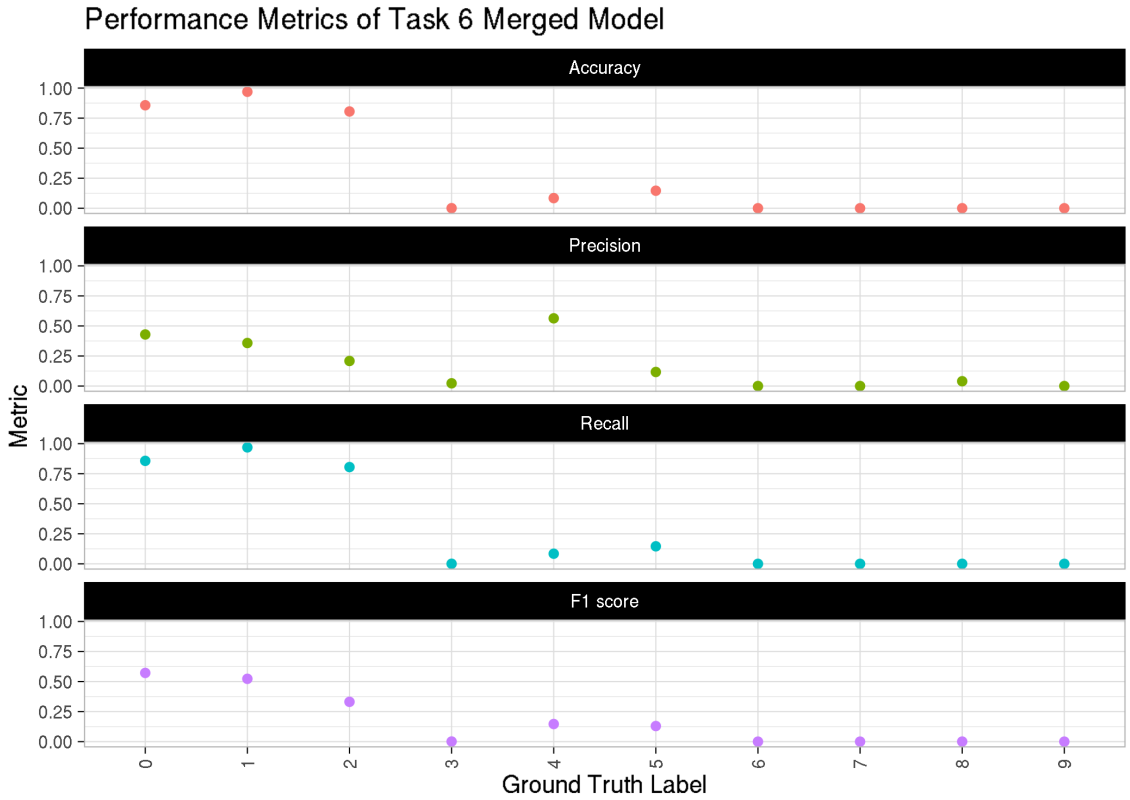


(b)

Figure S2: (a) Confusion Matrix and Performance Metrics for Task 5 Merged Model on entire EMNIST dataset

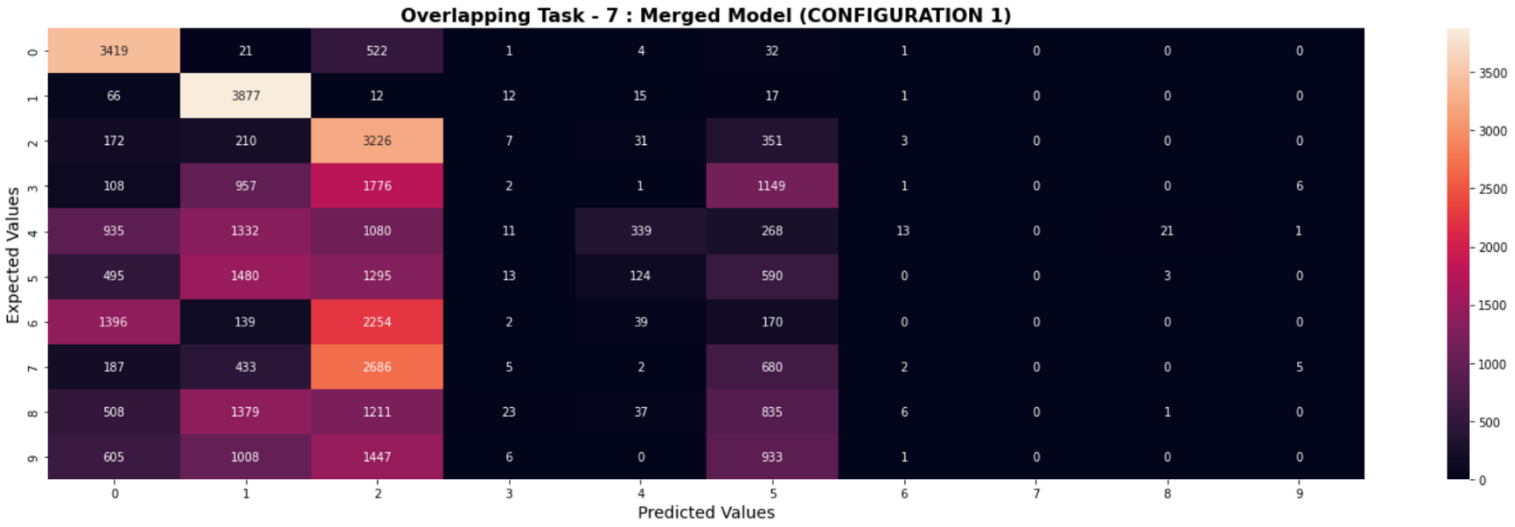


(a)

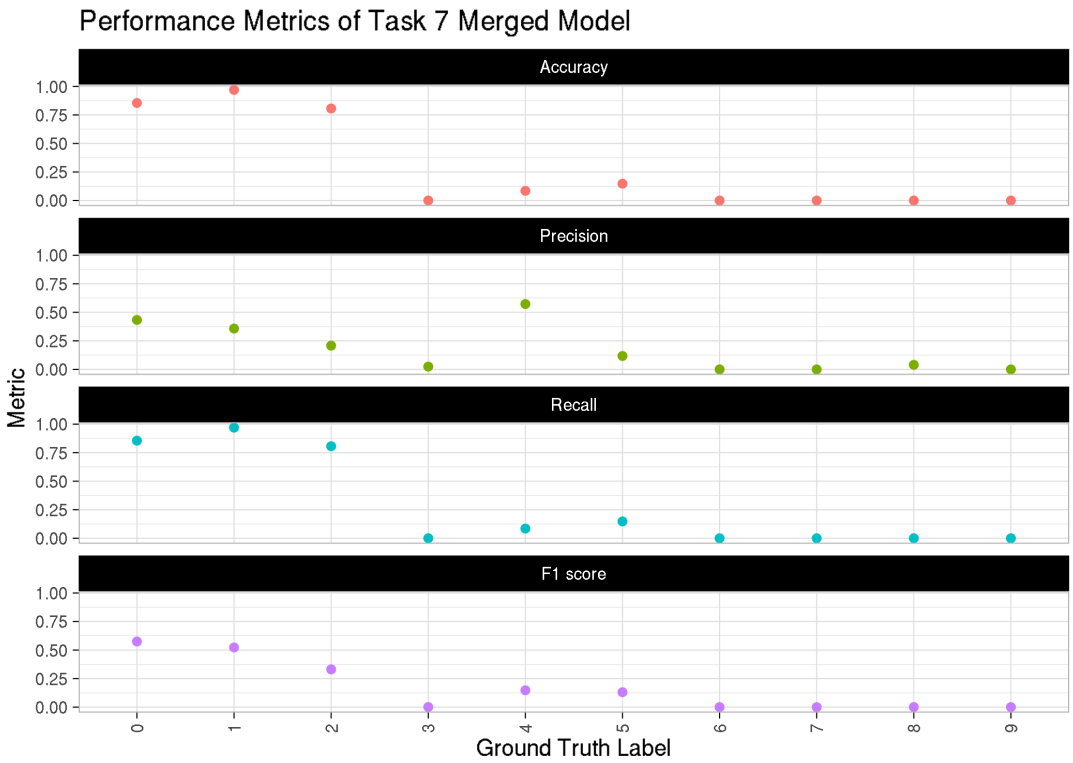


(b)

Figure S3: (a) Confusion Matrix and (b) Performance Metrics for Task 6 Merged Model on entire EMNIST dataset

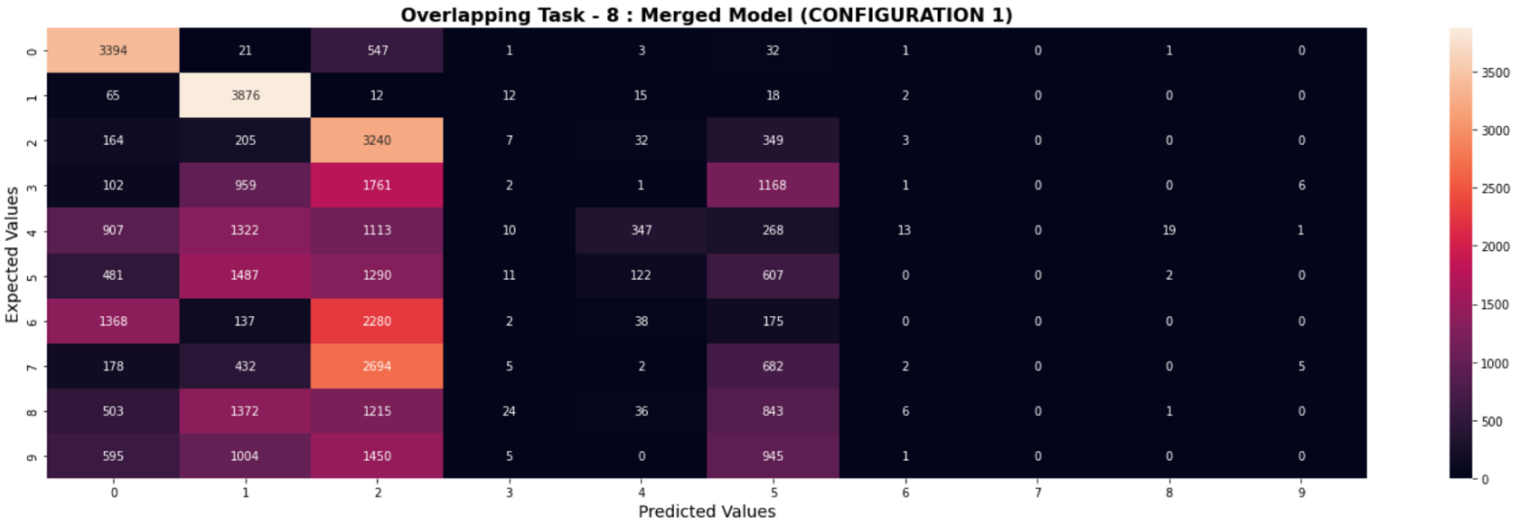


(a)

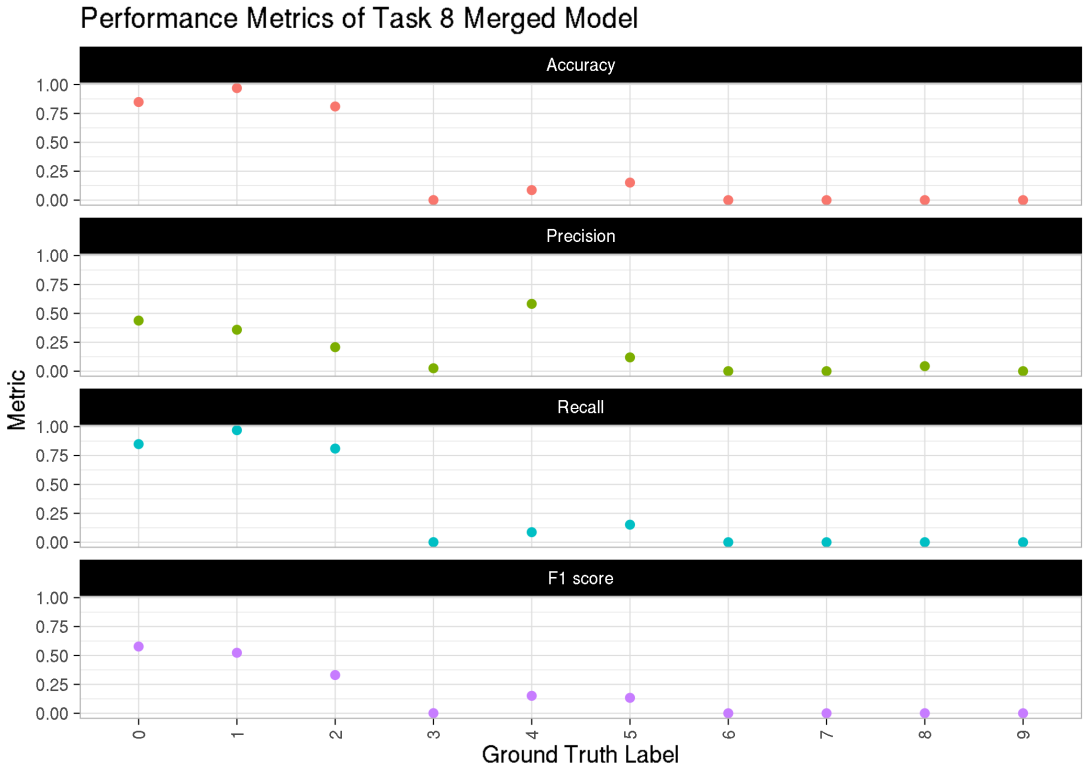


(b)

Figure S4: (a) Confusion Matrix and (b) Performance Metrics for Task 7 Merged Model on entire EMNIST dataset

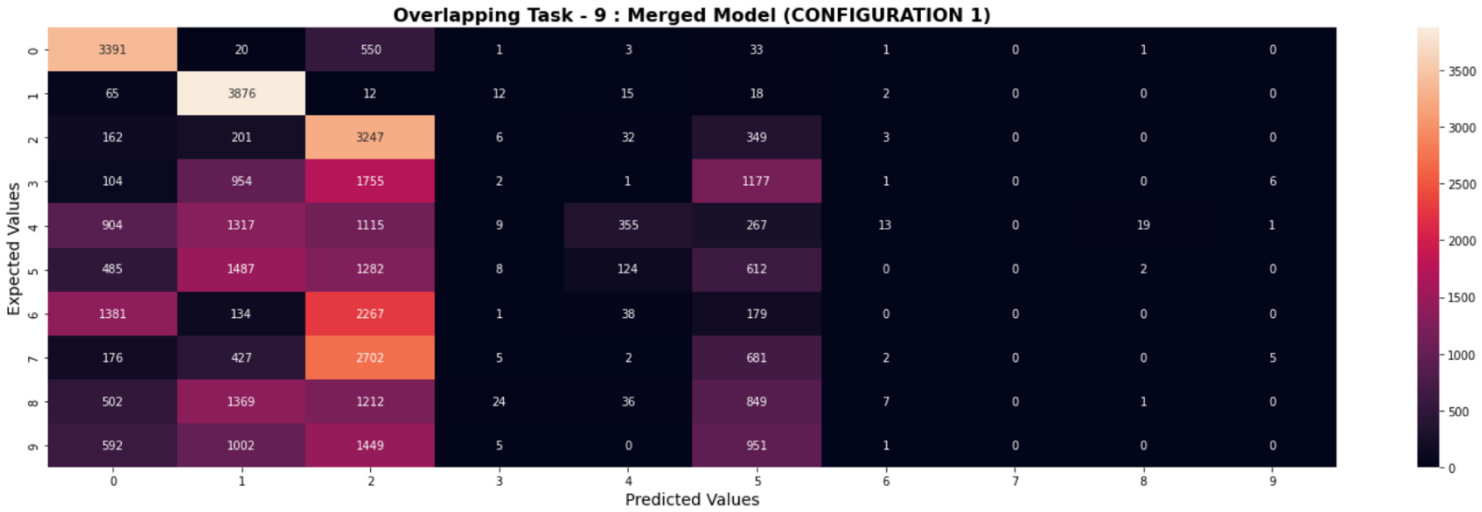


(a)

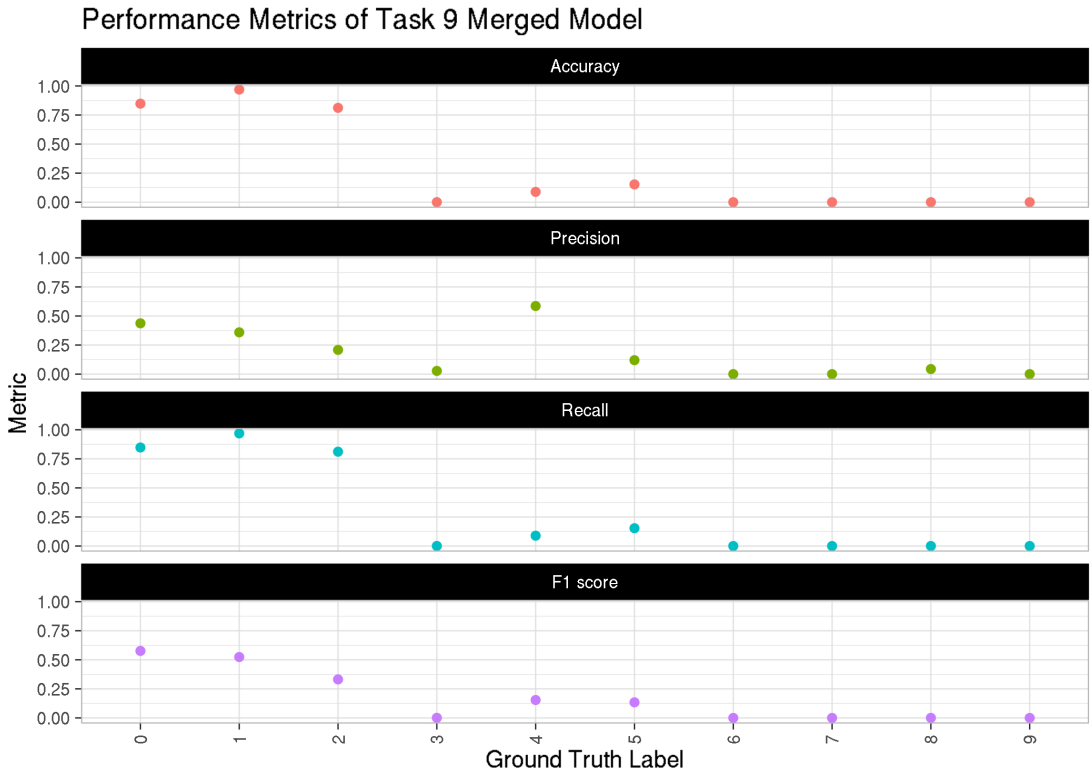


(b)

Figure S5: (a) Confusion Matrix and (b) Performance Metrics for Task 8 Merged Model on entire EMNIST dataset

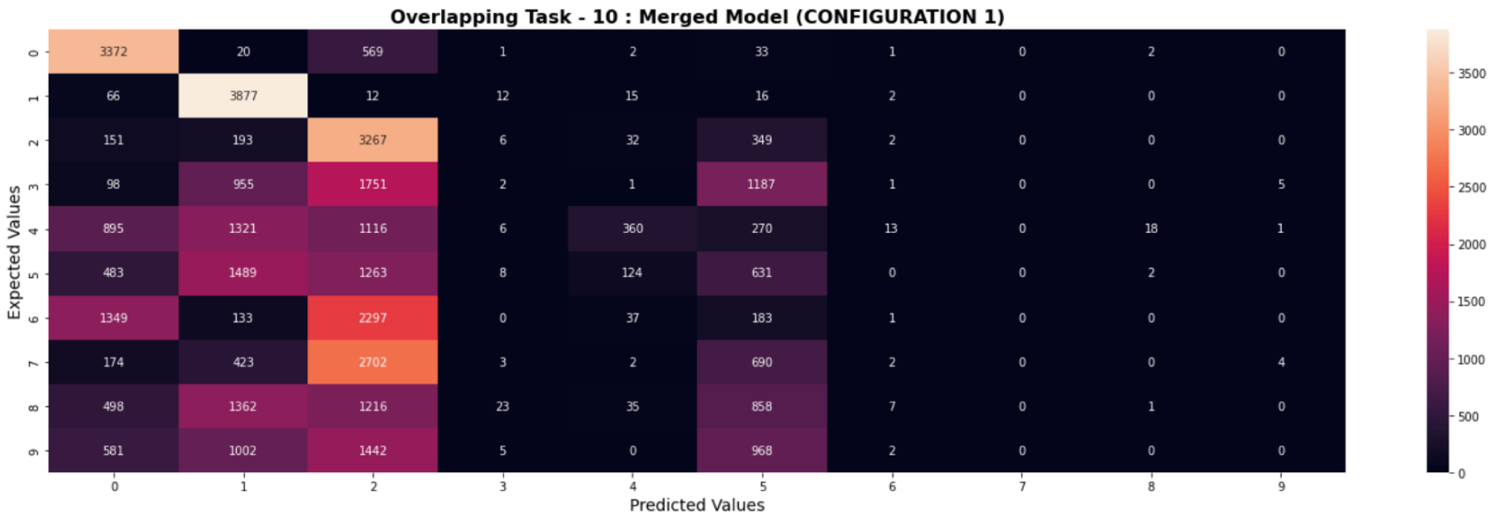


(a)

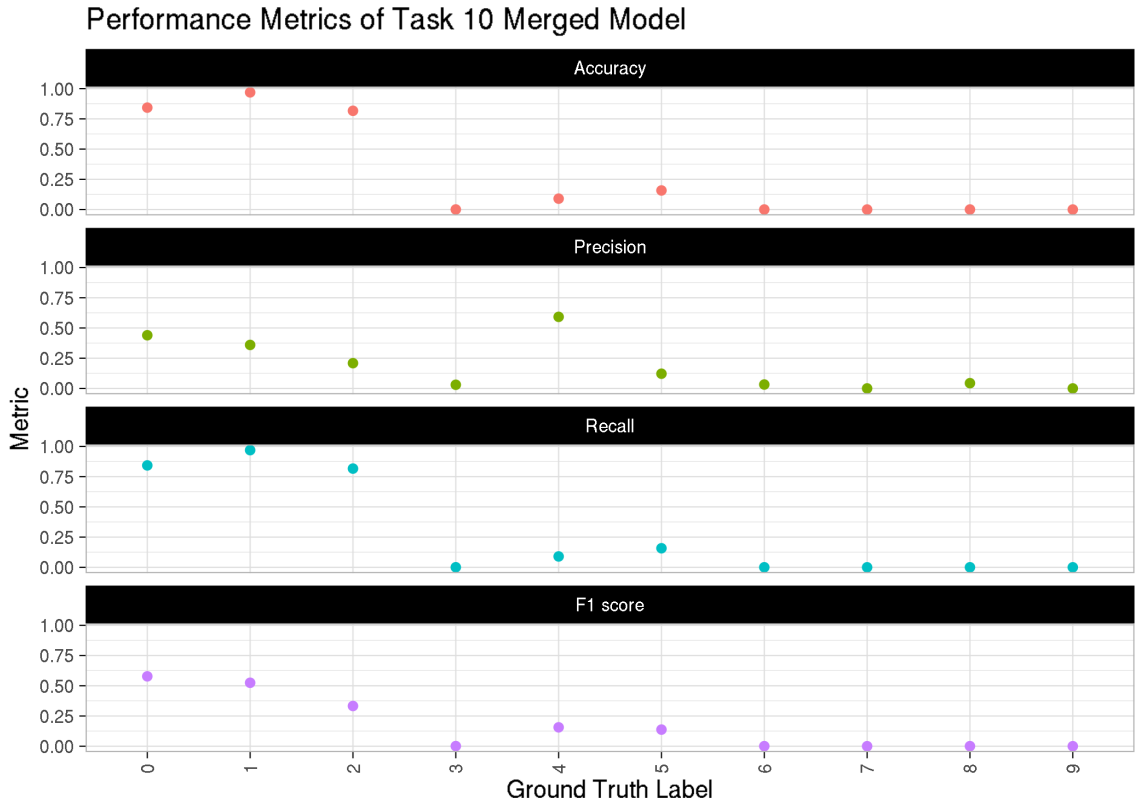


(b)

Figure S6: (a) Confusion Matrix and (b) Performance Metrics for Task 9 Merged Model on entire EMNIST dataset



(a)



(b)

Figure S7: (a) Confusion Matrix and (b) Performance Metrics for Task 10 Merged Model on entire EMNIST dataset